НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені
ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри _____
Сергій СТІРЕНКО
«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія» на

тему: «Система знаходження вразливостей

серверу»

Виконав:
студент IV курсу, групи ІО-62
Лавріненко Нікіта Тарасович _____

Керівник:
Доцент, кандидат технічних наук,
Верба Олександр Андрійович _____

Консультант з нормоконтролю:
Професор, доктор технічних наук
Сімоненко Валерій Павлович _____

Рецензент:
Посада, науковий ступінь, вчене звання,
Прізвище, ім'я, по батькові _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп’ютерна інженерія»

Освітньо-професійна програма «Комп’ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри _____

Сергій СТІРЕНКО

«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Лавріненку Нікіті Тарасовичу

1. Тема проєкту «Система знаходження вразливостей серверу», керівник проєкту Верба Олександр Андрійович, доцент, к.т.н., затверджені наказом по університету від «07» травня 2020 р. № 1081-с

2. Термін подання студентом проєкту 06 червня 2020 р.

3. Вихідні дані до проєкту: технічне завдання, науково-технічна література

4. Зміст пояснювальної записки: порівняльний аналіз існуючих програмних рішень, вибір засобів реалізації та опис отриманої системи

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо) :

    1. Функціональна схема – плакат
    2. Принципова схема – плакат
    3. Структурна схема – плакат

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Сімоненко В.П., професор		

7. Дата видачі завдання 01 вересня 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Затвердження теми роботи	01.09.2019-22.12.2019	
2	Вивчення та аналіз завдання	23.12.2019-20.03.2020	
3	Розробка архітектури та загальної структури системи	20.03.2020-01.04.2020	
4	Розробка структур окремих підсистем	01.04.2020-10.04.2020	
5	Програмна реалізація системи	11.04.2020-20.04.2020	
6	Оформлення пояснювальної записки	01.05.2020-23.05.2020	
7	Передзахист	24.05.2020-26.05.2020	
8	Захист	15.06.2020-20.06.2020	

Студент Нікіта ЛАВРІНЕНКО

Керівник Олександр ВЕРБА

АНОТАЦІЯ

Дана дипломна робота присвячена розробці системи знаходження вразливостей серверу, що призначена для збереження цілісної роботи серверу.

Система виконує пошук вразливостей наявних на сервері сервісів шляхом перегляду бази даних уже відомих вразливостей або їх веб-ресурсів.

Для реалізації системи було використано мову програмування Python з допоміжними бібліотеками.

ANNOTATION

This thesis is devoted to the development of a system for finding server vulnerabilities, which is designed to preserve the integrity of the server.

The system searches for vulnerabilities in the services that available on the server by viewing a database of already known vulnerabilities or their web resources.

The Python programming language with auxiliary libraries was used to implement the system.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	*ІАЛЦ. 467200.001* ВП	Відомість проєкту	1	
3	А4	*ІАЛЦ. 467200.002* ТЗ	Технічне завдання	3	
4	А4	*ІАЛЦ. 467200.003* ПЗ	Пояснювальна записка	57	
5	А4	*ІАЛЦ. 467200.004* Д1	Діаграма класів	1	
6	А4	*ІАЛЦ. 467200.005* Д2	Схема програми	1	
7	А4	*ІАЛЦ. 467200.006* Д3	Схема взаємодії програми з користувачем	1	
8	А4	*ІАЛЦ. 467200.007* Д4	Текст програми	17	

				ІАЛЦ. 467200.001 ВП	
	ПІБ	Підп.	Дата		
Розробн.	Лавріненко Н.Т			Відомість дипломного проєкту	Лист 1 / Листів 1
Керівн.	Верба О.А.				
Консульт.					КПІ ім. Ігоря Сікорського Каф. ОТ Гр. ІО-62
Н/контр.	Сімоненко В.П.				
Зав.каф.	Стіренко С.Г.				

Технічне завдання

до дипломного проєкту

на тему: «Система знаходження вразливостей серверу»

Київ - 2020 року

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

В даному технічному завданні розглянуто розробку системи знаходження вразливості серверу.

Область застосування: запобігання можливості несанкціонованого доступу до систем.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки служить завдання на виконання розробки системи знаходження вразливості серверу, затвердженою кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний Інститут».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка системи знаходження вразливості серверу.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами для розробки служать науково-технічна література з комп’ютерних технологій, публікації в періодичних виданнях, довідники з програмованих логічних інтегральних схем, публікації в Інтернеті за даним питанням.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розроблюваного продукту

- Сканування портів серверу на наявність утиліт.
- Знаходження вразливостей знайдених утиліт.
- Розробка графічного інтерфейсу для комфортного використання.

5.2. Вимоги до програмного забезпечення

- Unix-подібні операційні системи на базі ядра Linux.
- Python 3 і вище.

5.3. Вимоги до апаратного забезпечення

- Комп’ютер на базі процесору Intel Pentium 3 і вище.
- Оперативної пам’яті не менше в 1 Гбайт.
- Наявність Інтернет з’єднання.

Пояснювальна записка
до дипломного проекту

на тему: Система знаходження вразливостей серверу

Київ - 2020 року

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ ТА СКОРОЧЕНЬ

ОС	Операційна Система
LAN	(з англ. Local area network) обмежена комп’ютерна мережа, до якої підключені комп’ютери.
WAN	(з англ. wide area network) велика комунікаційна мережа, в яку входять LAN мережі.
IP address	(з англ. Internet Protocol address) ідентифікатор для підключення комп’ютера через мережу
TCP	(з англ. Transmission Control Protocol) протокол для передачі запиті з переглядом на помилки і повторного надсилання при її наявності.
SSL	(з англ. Secure Sockets Layer) криптографічний протокол для безпечної комунікації через мережу.
HTML	(з англ. Hypertext Markup Language) мова стилізації для веб-сайтів.
XML	(з англ. Extensible Markup Language) мова для збереження інформації у організованому виді.
CSV	(з англ. comma-separated values) мова для збереження інформації у організованому виді через роздільник “,” .
HTTP	(з англ.Hypertext Transfer Protocol

	) протокол передачі даних у HTML виді.
HTTPS	(з англ. Hypertext Transfer Protocol Secure) протокол для безпечної передачі даних у HTML виді через комп’ютерну мережу.
ICMP	(з англ. Internet Control Message Protocol) протокол який використовується для надсилання повідомлень про помилку запиту.
UDP	(з англ. User Datagram Protocol) протокол для передачі запиті без перевірки на помилки.
CVE	(з англ. Common Vulnerabilities and Exposures) унікальний ідентифікатор для вразливості.
API	(з англ. application programming interface) інтерфейс, який забезпечую взаємодію між різними програмами.
ООП	Об’єктно-орієнтоване програмування – парадигма програмування, яка класифікує об’єкти і їх взаємодію.

ВСТУП

На даний момент популярність інтернету набуває свого піку, що майже кожен використовує його в той чи іншій мірі, тому кількість онлайн сервісів зростає. Для будь-якого навіть маленького бізнесу наявність його опису в інтернеті є необхідністю. Відповідно до цього кількість серверів, які утримують ці сервіси теж зростає, підвищується їх потужність і складність. І з’являються пробіли в безпеці, які можуть використовувати для несанкціонованого доступу до систем, або для порушення цілісності системи в цілому. З цього моменту стають актуальними системи для пошуку вразливостей серверу, які шукають ці пробіли і дають оцінку проблемі. Але в цих системах є один мінус — поновлення бази даних новими ураженнями і неповна оцінка проблеми, що призводить до інколи необ’єктивного представлення проблеми і пропуску наявності пробілу в безпеці.

Актуальність теми

Через зростання популярності онлайн сервісів і їх великому різновиді, виникає потреба в розробці системи для тестування серверів на вразливість, для забезпечення безпеки сервісу.

Мета і задачі

Метою роботи є розробка нової системи пошуку вразливостей серверу, яка буде знаходити ураження відповідно до сервісу, з можливою наявністю кодом для злому цього сервісу, з швидким поновленням бази даних вразливостей і оцінки проблеми.

Для реалізації даної системи були поставлені наступні задачі:

- Провести аналіз існуючих систем знаходження вразливостей серверу;
- Визначитись з методом сканування сервісів;
- Обрати бази даних вразливостей;

- Дослідити та проаналізувати оцінки вразливостей, відповідно до їх степені ураження.

Практичне значення

Даний метод полягає у збільшені звичайної бази даних вразливостей шляхом використання аналізу проблеми з багатьох баз даних, що дає більш об’єктивну оцінку ураження, а також більшу кількість інформації про вразливість.

РОЗДІЛ 1

ВРАЗЛИВОСТІ. ЇХ ВПЛИВ НА СЕРВЕРНІ СИСТЕМИ.

1.1 Мета пошуку вразливостей. Типи вразливостей. Історичні приклади злому. Навіщо кожній компанії система пошуку вразливостей.

1.1.1 Розгляд понять.

Сервер – це комп'ютер, який надає дані іншим комп'ютерам. Він може обслуговувати передачу даних в системах локальної мережі (LAN) або широкосмугової мережі (WAN) через Інтернет.

Брандмауер – це система мережевої безпеки, яка контролює вхідний та вихідний мережевий трафік на основі заздалегідь визначених правил безпеки. Брандмауер зазвичай встановлює бар'єр між надійною внутрішньою мережею та ненадійною зовнішньою мережею.

1.1.2 Мета пошуку вразливостей

У кожного сервера, який забезпечує певний онлайн сервіс бізнесу з’являються проблеми в безпеці, що погрожують цілісності цього сервісу. Зловмисники можуть використати дані пробіли в безпеці для того щоб викрасти специфічні дані з серверу. Використовуючи дану інформацію не в хороших намірах для бізнесу. Зазвичай, ці дані можуть складатись з паролі та логінів користувачів, які використовують цей онлайн сервіс, а зайшовши на платформу цього бізнесу, можна вже знайти більш чутливу інформацію: приватні фото, відео, листування і потім шантажувати цими даними. Це може бути не тільки медіа інформація, а й транзакції перерахунків, банківські рахунки. Зараз набувають популярності сервіси інтернет банкінгу, а також і мобільного банкінгу. Таких сервісів стає все більше і більше, тому наявність вразливостей теж зростає. Зловмисники використовують багато тактик для проникнення на сервер, залучаючи навіть персонал цього сервісу, який нічого не підозрює. Їх головною ціллю являється

викрасти інформацію, але це не завжди так. Поділимо всі цілі злому відповідно до інтересів хакеру:

- Викрадення інформації з метою шантажування
- Порушення роботи сервісу
- Ради інтересу, забави
- Маніпулювання банківськими транзакції з метою збагачення
- Ради хакерського ідеалу, так би мовити за справедливість, виступаючи проти уряду, або релігійних груп
- Тестування, для усунення ураження

Як видно більшість цілей направлені на збагачення, хоч і треба відмітити направленість хакерів на інтерес ідеалу чи забави, тому що такі зловмисники більш наполегливіші і захопленні своєю роботою. Знаючи інтереси хакерів і спеціалізацію сервісу можливо передбачити їх цілі на сервері і завчасно запобігти проникненню зі сторони зловмисника.

Мішенню для зловмисників може стати будь-який онлайн сервіс, але потреба у зломі може і не бути, якщо наприклад, це сайт звичайної газети чи блогу, то потреба у викрадені цінної інформації відпадає. Зазвичай такий тип сервісів рідко чіпають, але можливість його порушення зацікавить конкурентів і вони наймуть зловмисника для того , щоб скомпрометувати надійність цього бізнесу і показати що вони не найкращі. Але більш прибутковим для хакерів все ж таки буде проникнення в бази даних банків, або онлайн сервісів з великою кількістю клієнтів і отримання доступу до цінної інформації.

1.1.3 Типи вразливостей системи

Вразливість – пробіл в безпеці, який можна використати для не запланованої роботи сервісу і його порушення. Тобто це дірка системи, яку можна використати для викрадення інформації або для подальшого просуванню по системі. Зловмисники шукають саме такі вразливості, для несанкціонованого

входу в систему. Розглянемо, які типи вразливостей існують на даний момент описані в книзі “Penetration Testing”[1]:

- Вразливість програмних забезпечення, тобто знаходження помилок в коді програми, так звані баги, які ведуть до не коректної роботи сервісу. Неправильно виділена пам’ять для програми чи її переповнення чи порушення логіки роботи веде до експлуатації цієї програми для отримання більших прав для запуску інших сервісів, в яких може бути така ж проблема, до тих пір поки зловмисник не отримає могутні root права і зможе роботи все що захоче в даній системі.
- Вразливість операційної системи, тобто помилка в операційній системі чи не запущений сервіс, який уникає помилок. Зазвичай ця вразливість проявляється, за недбалості чи не хватки досвіду системного адміністратора. Чи надавання завищених привілеїв програми, яка того не потребує і яку можна використати для запуску інших програм з цими привілеями
- Брандмауер вразливості, брандмауер захищає сервіси, які стоять на сервері від зовнішніх атак і коли в нього відбувається збій чи в ньому знаходять вразливість, то і під наступну ціль попадають і самі сервіси. Тобто його може навіть і не бути , або він буде не запущений, через халатність системного адміністратора. Без брандмауеру зловмисник спокійно може подорожувати по мережі.
- Вразливості Веб серверу, вразливості зі сторони веб сервісу, які надають доступ до системи чи до бази даних, зазвичай такі проблеми в безпеці виникають через неправильним дизайном цього сервісу або невдалою обробкою помилок на сервері, що веде до фатальних помилок уже в системі.
- Вразливості TCP/IP, це отримання доступу до тої частини мережі, яка не призначена для використання клієнтом, а для адміністративної роботи. І це суттєва вразливість так як зловмисник без проблем може заходити в

робочу мережу і отримати конфіденційну інформацію с боку системи, які мережі ще є, хто підключений до мережі і т.д.

- Вразливість Wi-fi, тобто в компанії налаштована Wi-fi мережа і зловмисник знаходячись в радіусі знаходження мережі може отримати доступ до неї шляхом злому, або вона може бути навіть без захисту. Ціль даного ураження залізти далі в мережу компанії для отримання доступу до комп'ютерів чи серверів компанії.

1.1.4 Історичні приклади злому

Щоб краще оцінити збитки компаній від злодіянь хакерів буде приведено пару прикладів відомих компаній, які допустили ураження зі сторони зловмисників і вони використали дану вразливість для отримання інформації для шантажу чи компрометування чи для порушення їх роботи.

Таблиця 1.1: Наслідки компаній після викрадення інформації

Компанія	Наслідки
Abode	Викрадено біля 3 млн зашифрованих кредитних карт з логінами і паролями. Було виплачено 1.1 млн доларів для погашення судового боргу за розкриття особистих записів клієнтів
LinkedIn	Викрадено біля 165 млн аккаунтів, через халатність персоналу на безпеку. Цифри по виплаті не були знайдені, тому судити про збитки ми можемо тільки теоретично
Yahoo	Викрадено 3 млрд аккаунтів користувачів і її нарекли як найбільша інформаційна діра в історії. Там була інформація про ім'я, адреси, телефони і т.д. І щоб покрити дані втрати, компанія виплатила 350 млн доларів
Sony	Викрадено 77 млн аккаунтів. Інформацію про банківські карти, логіни і паролі. Компанією було виплачено близько 15 млн доларів для виплати користувачам.

Таблиця 1.1: Наслідки компаній після викрадення інформації

Компанія	Наслідки
Target	Викрадено 110 млн аккаунтів з банківською інформацією і особистими даними. Обійшлось це компанії близько 18 млн доларів.

В табл. 1.1 було наведено компанії і їх витрати на відновлення після хакерських атак, інформація взята з веб-ресурсу[2], суми дуже великі і щоб не встрявати в такі проблеми, компаніям треба перевіряти та покращувати безпеку шляхом імітуванням зловмисних атак.

1.1.5 Навіщо кожній компанії система пошуку вразливостей

Коли молода компанія стає популярною її прихильників більшає, як і противників. Це призводить до того, що на таку компанію почнуть спокушатися зловмисники для отримання вигоди або ради цікавості і випробування своїх сил. В той час компанія може і не задумуватися про таке, поставивши брандмауер і забувши про хакерів, але не все так просто і є купа способів його обходити. І коли в компанії зловмисник знайшов дірку, тільки тоді починають задумуватися про такі утиліти, які допоможуть у боротьбі з хакерами та проаналізують систему на наявність пробілів в безпеці шляхом сканування на вразливості. В даному пункті ми розберемо тему електронного ресурсу[3], чому ж серверу потрібна система для пошуку вразливостей:

- Виявлення вразливостей в діапазоні мережі компанії. Періодичне сканування підтвердить безпеку мережі і сервісів, які там налаштовані, що всі утиліти останньої версії або версії, яка не була вразливою для хакерських атак, для цього треба своєчасне оновлення бази даних вразливостей для уникнення версій, які підпадають під категорію уражених, або блокування процедури, яка вважається вразливою.

- Підтвердження того, що останні зміни в сервісі були безпечними і не створили пробілів в безпеці, які можуть використати зловмисники для проникнення в систему.
- Перевірка системних конфігурацій, наприклад був наданий доступ до тої частини мережі, яка виявилась адміністративною і не призначеною для клієнта. Що всі права на утиліти і доступи налаштовані згідно роботи системи.
- Виявлення вразливостей у інших сервісів, які б могли надати доступ до основного. Хакер може уразити допоміжний сервіс, який буде знаходитись на одній машині з основним і отримати права на користування ним.
- Переконання клієнтів, що ваша система в безпеці і під наглядом. Це дає змогу клієнту довіритись такій компанії більше, а отже і популярність збільшиться і кількість клієнтів теж, що приведе до більшого доходу такого сервісу.

1.2 Аналіз існуючих систем пошуку вразливостей

Проаналізувавши системи пошуку можна дізнатися, який з функціоналу є найбільш ефективний для реалізації пошуку і оцінки вразливостей, який з функціоналу полегшує роботу роботу. Дізнавшись про дану інформацію буде вибрано найкращі реалізації з різних систем.

1.2.1 Nikto

Огляд функціоналу

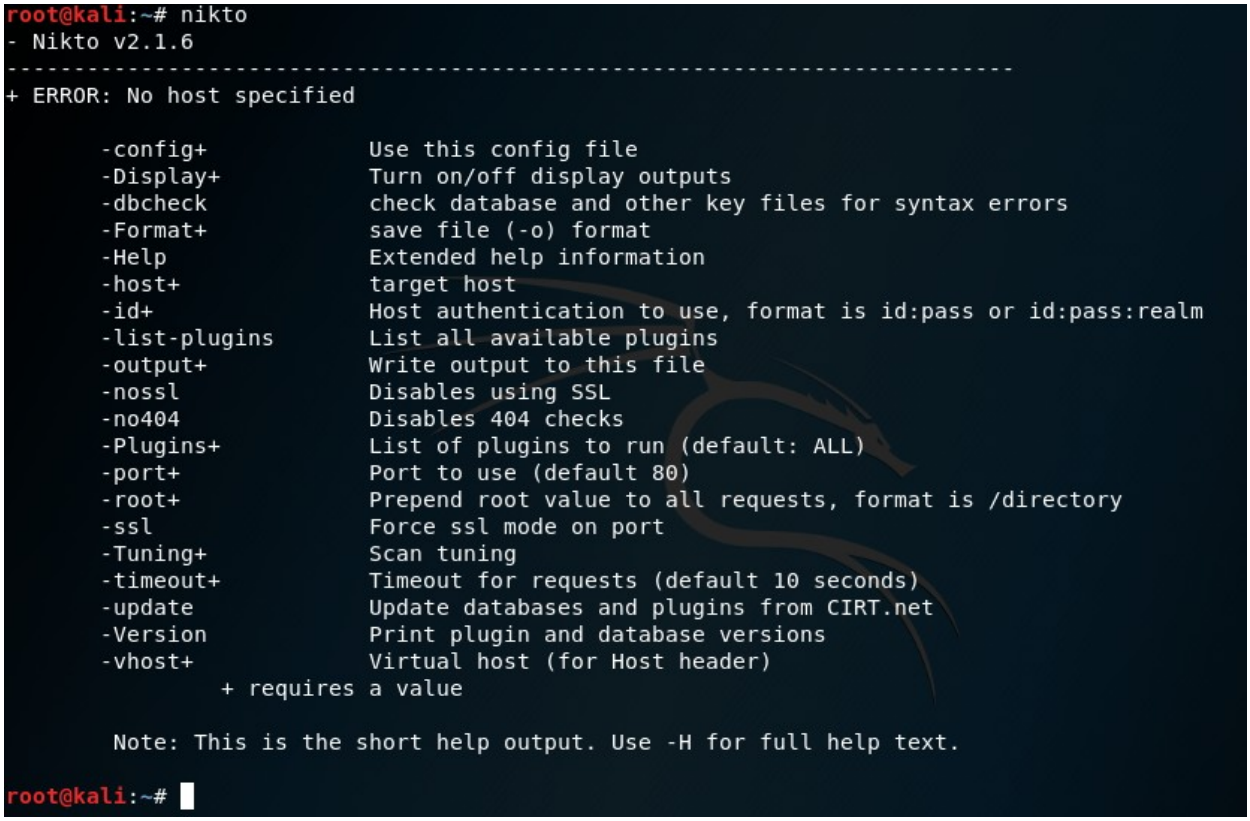


Рисунок 1.1 — Інтерфейс утиліти Nikto

На рисунку 1.1 представлена утиліта Nikto[4] для пошуку вразливостей серверу, за допомогою якої проходить сканування на ураження HTTP і HTTPS сервісів, в плані і в використанні саме цих портів вона одна з найкращих зазначено на сайті [4]. Програма перебирає можливі сторінки сайтів, до яких можливо адміністратор забув закрити доступ, сповіщає про незвичайні заголовків в запитах.

Переваги та недоліки

Переваги:

- Швидкий перебір можливого доступу на закриті домени
- Робота з SSL
- Перевірка версій на серверних компонентах
- Збереження рапорту про знайдену інформацію в форматах XML, HTML, NBE or CSV

Недоліки:

- Консольний графічний інтерфейс
- Робота тільки з HTTP і HTTPS портами
- Немає оцінки вразливості, тільки їх можливу наявність

Для свого спектра програма достатньо хороша і реалізація знаходження вражає, але малий функціонал для забезпечення безпеки всього серверу. Для реалізації власної системи пошуку буде взято принципи обробки HTTP і HTTPS в утілити Nikto.

1.2.2 Rapid7 Nexpose

Огляд функціоналу

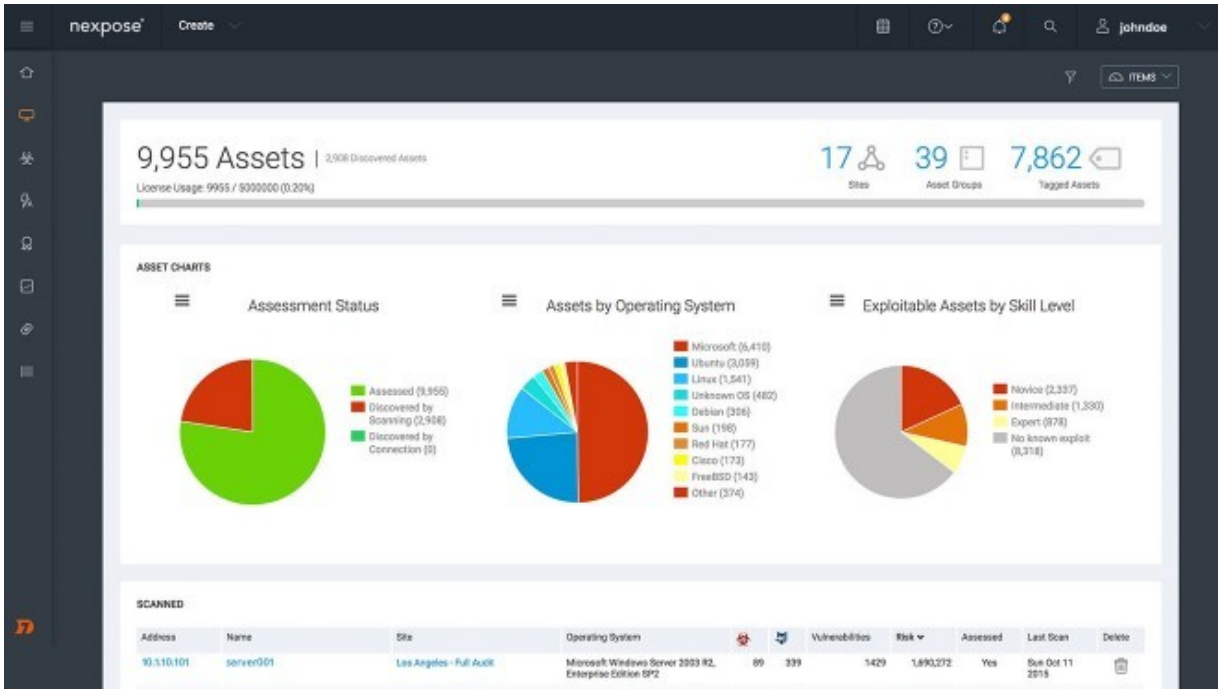


Рисунок 1.2 − Інтерфейс програми Rapid7 Nexpose

На рисунку 1.2 зображено Rapid7 Nexpose[5] локальний інструмент для пошуку пробілів в безпеці системи, шляхом сканування мережі, утиліта застосовує унікальну оцінку ризику і робить детальне порівняння уражень. В створеному графічному інтерфейсі легко зрозуміти положення кожної з функцій системи, а графіки статистичних даних відразу дають зрозуміти стан серверу. Але кількість

уражень утиліта знаходить на порядки менше, простої перевірки в даній системі не достатньо.

Переваги та недоліки

Переваги:

- Об’єктивна оцінка вразливості
- Легко-зрозумілий інтерфейс
- Статистичні дані про стан серверу

Недоліки:

- Неповний набір вразливостей, які може опрацювати дана система
- Статистичні дані не завжди збігаються з реальністю, через попередній пункт
- Велика ціна за користування

Дана утиліта легка в користування, навіть користувачу, який нічого не розуміє в безпеці мережі і представлення оцінки вразливості теж дає змогу швидше зрозуміти стан серверу, але неповне знаходження вразливостей не дає точного представлення про ситуацію на ньому, а також велика ціна підписки, яка потрібна для користування.

1.2.3 Metasploit

Огляд функціоналу

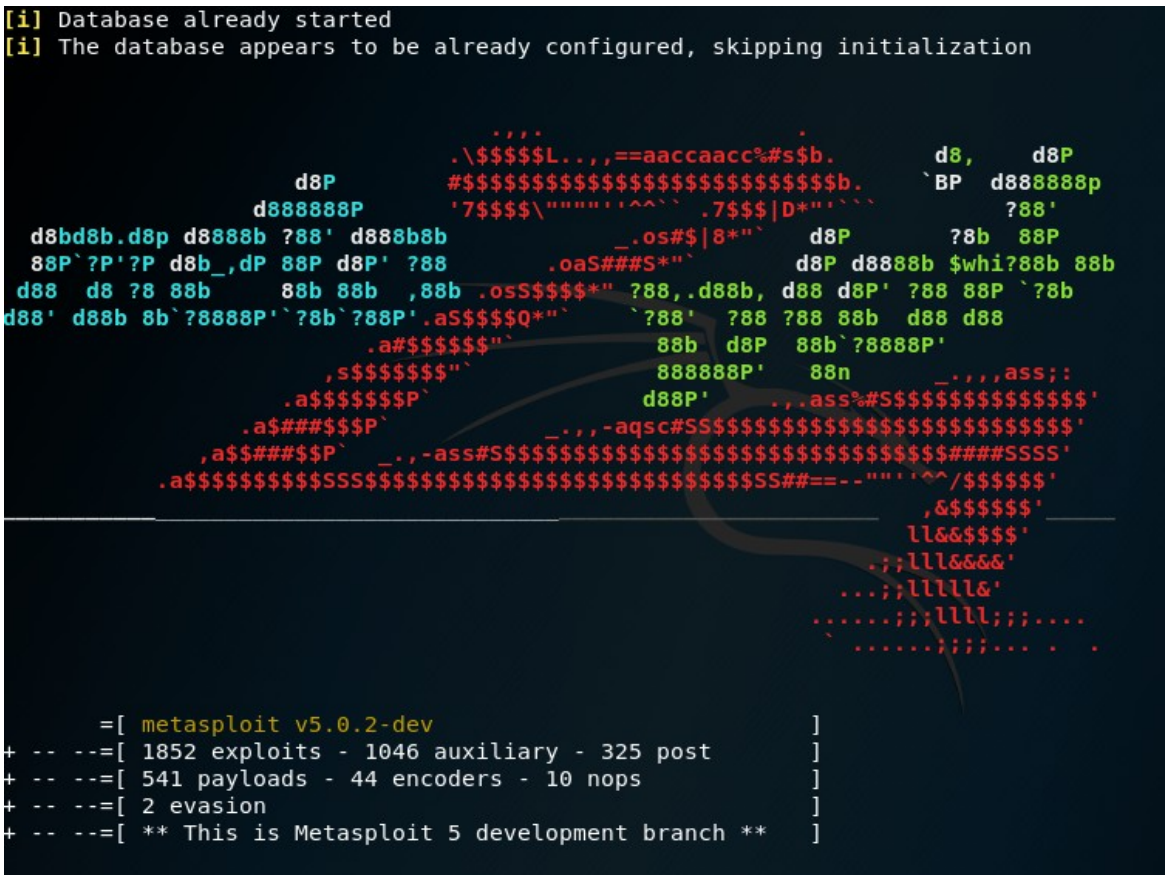


Рисунок 1.3 − Інтерфейс програми Metasploit

На рисунку 1.3 зображена популярна програма для пошуку вразливостей і створення програм, які уражають систему – Metasploit[6]. За її горами велика кількість відкритого і доступного коду для використання вразливостей, також в даній утиліті присутній їх аналіз і доступна покрокова експлуатація. Для нових користувачів не відразу буде зрозуміло, як нею користуватися, адже інструментів для тестування в ній багато і яку потрібно вибрати і для чого, треба вміти. А для клієнтів, які вже мають досвід роботи с даною програмою, то це найкращий інструмент для тестування, але через його популярність в тестуванні всі ураження, які знаходить Metasploit вже були виправлені, тому актуальність в даному ресурсі знецінюється.

Переваги та недоліки

Переваги:

- Великий функціонал тестування
- Вже створені експлуатаційні коди для вразливостей, тобто прямо з Metasploit можливо здійснити в систему чи проникнути в систему чи порушити роботу сервісу.
- Аналіз вразливостей. Типи різного характеру, для оцінювання пробілів в безпеці, які описані в п.1.1.3.

Недоліки:

- Інтерфейс. Інтерфейс з консолі не зручний для використання, а для використання утиліті з графічним інтерфейсом потребує купівлі підписки на Metasploit Pro.
- Популярність. Через велику популярність в тестуванні деякі ураження важко знайти.

Як для безкоштовного використання Metasploit один із найкращих сканерів і його можливість до використання коду для порушення роботи сервісу, це відбувається через підтримку великої кількості людей, які доповнюють базу даних вразливостей. Для його використання треба багато практики сканування і експлуатації, будь-який клієнт може дописати новий модуль для вразливостей, хоч для цього і потрібно знати Ruby.

ВИСНОВКИ ДО РОЗДІЛУ 1

Вразливість називають слабкість або погану реалізацію в системі, яку можна використати для злому сервісу. Основні мотиви для злому сервісу :

- Викрадення інформації з метою шантажування
- Порушення роботи сервісу
- Маніпулювання банківськими транзакції з метою збагачення
- Ради хакерського ідеалу, так би мовити за справедливість, виступаючи проти уряду, або релігійних груп
- Тестування, для усунення ураження

Компанії несуть великі збитки від спричинених зловмисниками діяннями, це може порушити роботу такого сервісу, а також компрометувати його як не надійний сервіс і користувачі не зможуть йому довірити свою особисту інформацію, що ставить під загрозу популярність даного бізнесу.

Проведено аналіз уже існуючих систем пошуку вразливостей і виділено переваги і недоліки кожної для створення кращого функціоналу, який є в даних системах.

РОЗДІЛ 2

ПРОЕКТУВАННЯ СИСТЕМИ ЗНАХОДЖЕННЯ ВРАЗЛИВОСТЕЙ

2.1 Огляд предметної області

2.1.1 Бази даних вразливостей

База даних[7] – інформація, яка організована певним чином для легшої її маніпуляції, тобто для запису, читання, поновлення. Дані організовані в виді таблиці, яка розбивається на колонки та стовпчики, для швидкого пошуку потрібної інформації.

Для документування наявності ураження в програмах чи сервісах, були створенні бази даних з описом вразливостей, в яких описується тип ураження, його значність, спосіб проникнення. Є офіційні бази даних і не офіційні, тобто відкриті, де кожен охочий може викласти свої способи експлуатації сервісів системи. Деякі бази представлені в не звичайному форматі – через веб-сайт, це зроблено для того щоб забезпечити безпеку цих баз, а деякі надають прямий доступ. Кожній вразливості, яку офіційно визнали, тобто компанія продукту виклала в базу, що в них виник даний пробіл в безпеці, чи довірений представник даної компанії перевірив, що вразливість дійсно існує і заніс її в базу даних. Їй надають певний номер Common Vulnerabilities and Exposures (CVE)[8], – це ім’я вразливості, яке було стандартизоване відповідно до типу та загрози яку вона несе. CVE допомагає швидше віднайти ураження по так би мовити адресі. З одної сторони це добре для компаній, тому що вони подивились в базу даних і виправили нову вразливість, яка з’явилася в базі, все швидко і автоматично. Але з іншої сторони зловмисникам, теж стало легше шукати вразливість і тут палка двох кінців. Цю проблему вирішує саме система пошуку вразливостей, тому що вона отримує нові данні миттєво, до того як хакер встигне її використати.

2.1.2 CVEDETAILS

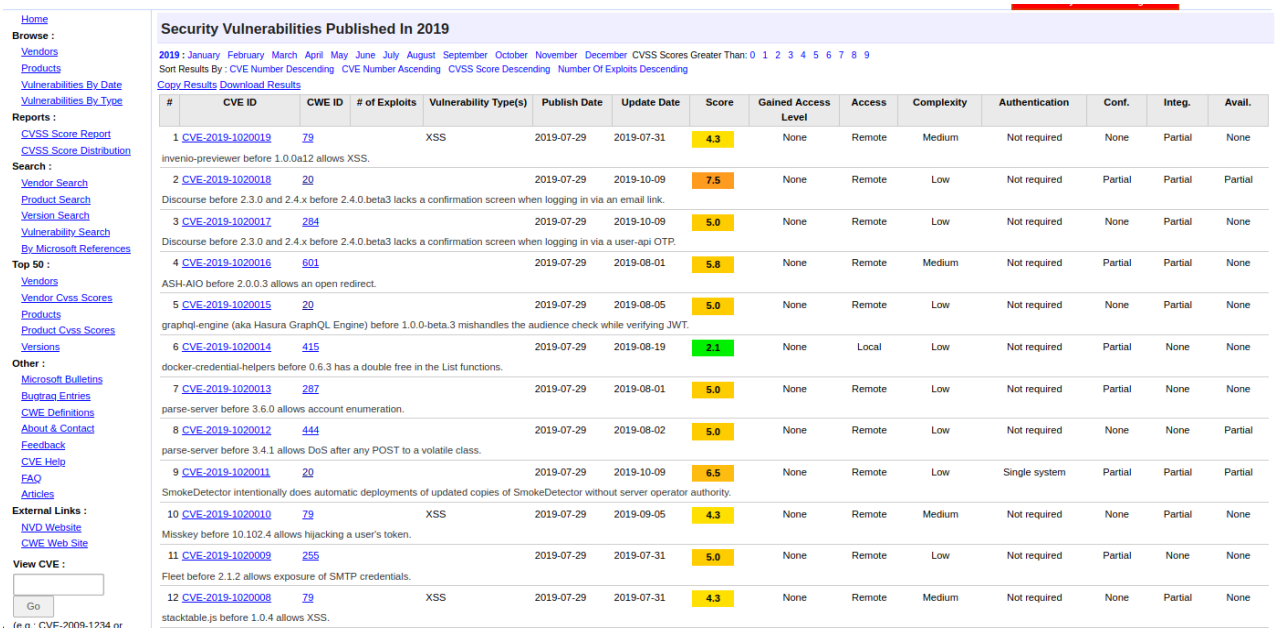


Рисунок 2.1 − Інтерфейс сайту www.cvedetails.com

На рис. 2.1 зображено веб-ресурс cvedetails[9] – це веб представлення бази даних National Vulnerability Database, де знаходяться вразливості, які були офіційно перевірені. Як бачимо з рис. 2.1 інформація про вразливість ділиться на такі колонки:

- CVE, CVE ID
- Vulnerability type(s).Тип вразливості.
- Publish Date і Update date. Дата публікації і поновлення.
- Score. Оцінка вразливості.
- Gained Access Level. Отриманий рівень доступу/
- Access. Тип доступу.
- Complexity. Складність у використані ураження.
- Authentification. Наявність аутентифікації.

На даному ресурсі можна з легкість віднайти вразливість продукту відповідно до її версії, а це один із важливіших аспектів пошуку ураження системи. Багато систем використовують саме цю базу даних для своєчасної перевірки продукту.

2.1.3 EXPLOIT-DB

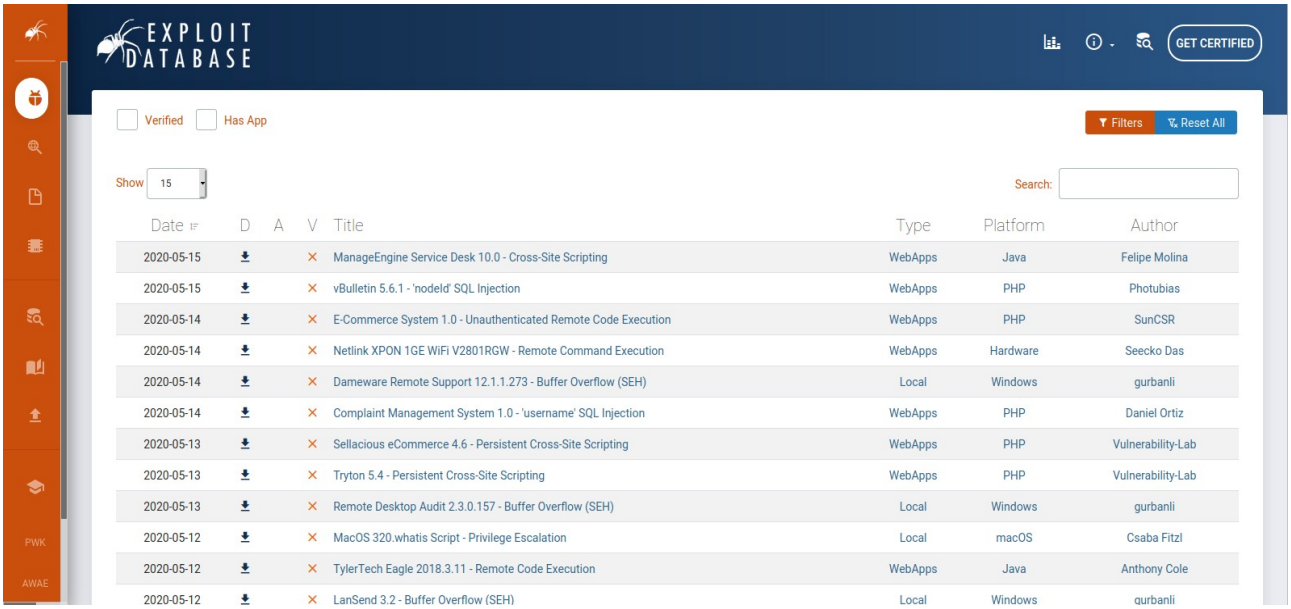


Рисунок 2.2 − Інтерфейс сайту www.exploit-db.com

На рис. 2.2 зображено веб-ресурс exploit-db[10] – це веб представлення бази даних exploit-db, де розмішені, як перевірені так і не перевірені вразливості та їх експлуатаційне пояснення. На цьому ресурсі можливо знайти використання ураження, тобто код для її експлуатації. Вона ділиться на такі колонки:

- Date. Дата публікації.
- D. Загрузити.
- V. Перевірене чи не перевірене ураження.
- Title. Заголовок.
- Type. Тип продукту.
- Platform. Платформа, на якій працює ураження.
- Author. Автор даної публікації.

На даному ресурсі представлені багато публікацій, які не перевірені, адже автором даної публікації може бути будь-хто, але в основному експлуатація вразливостей працюють, просто не вистачає людей для того, щоб це перевірити. Тому це може бути чудовим інструментом для пошуку не перевірених вразливостей.

2.2 Етапи для пошуку вразливостей

2.2.1 Виділення етапів знаходження і ідентифікації вразливості

Для здійснення проникнення в систему чи для того, щоб порушити сервіс зловмисники придержуються певного плану для досягнення своїх цілей, тому ми будемо їх імітувати для знаходження пробілів в системі, які приведуть до її експлуатації. Цей план буде складатися з таких етапів, які добре описані в книзі “Penetration Testing: A Hands-On Introduction to Hacking”[1]:

- Знаходження мереж, які в подальшому будуть атаковані.
- Сканування портів серверу, на наявність сервісів, які можуть бути атаковані.
- Аналіз сервісів, які знайшли на попередньому етапі на наявність вразливостей.
- Аналіз вразливостей і їх оцінка на можливе проникнення в систему чи порушення його роботи.

2.2.2 Знаходження мереж

На даному етапі зловмисник вивчає інформацію про компанію і знаходить її мережі, це можу бути мережі, як в інтернеті так і Wi-fi біля офісу компанії головною ціллю є дістатися до мережі компанії. Хакер навіть може пробратися в офіс компанії та отримати доступ до комп’ютера одного із працівників компанії. Цей етап включає в себе ціль розпізнавання мережі компанії і вже на даному етапі можуть виникнути проблеми. Наприклад сервер використовує проксі, або Wi-fi в офісі не під’єднаний до основної мережі чи охорона в компанії добре опрацьовувана, що його не пустять нікуди. Але є способи обійти і навіть ці проблеми за допомогою соціальної інженерії. Шляхом надсиланням співробітникам компанії листів з фішинговими сайтами, на яких можна отримати дані з робочого комп’ютера або навіть доступ до нього.

2.2.3 Сканування портів серверу.

Кожний сервіс, який використовує комп’ютерну мережу займає порт, тобто можна сказати, що для окремого процесу пов’язаного з мережею виділяється порт для її використання і при TCP/UDP запитах порт вставляється як ідентифікатор додатку в заголовок пакетів для нього виділено місце в 16 бітів, тому і діапазон портів від 0 до 65535. Тому нам так важливо дізнатися, який сервіс використовує той чи інший порт. Для цього треба просканувати сервер на наявність відкритих портів, щоб дізнатися про сервіс, який там знаходиться і протестувати його на наявність вразливостей.

Сканування портів[11] - це процес, який надсилає запити певному діапазону адрес портів сервера, з метою пошуку активного порту. Нас цікавлять саме активні порти, адже з відкритого порту ми можемо ідентифікувати доступні служби, що працюють на сервері чи яка Операційна система запущена, чи наявність брандмауера. Але дізнатися активні порти не завжди так легко, адже брандмауер може бути налаштований так, що буде зберігати ваш IP адрес, збільшить проміжок часу між відповідями на ваші запити, або взагалі заблокує вас, тому метод простого перебору портів не завжди спрацьовує. Тобто треба реалізувати таке сканування, яке б дало змогу обходити брандмауер.

2.2.4 Види сканувань.

Існує декілька різних методів сканування портів. Нижче перераховано кілька з багатьох методів, які добре описані в книзі “Hacking: The Art of Exploitation”[11] та те, як вони працюють:

- Сканування Ping – надсилання запиту ICMP до порту сервера, якщо сервер відповідає, то порт працює.

  Переваги:

  - Швидкість.
  - Покривання великої частини мережі.

  Недоліки:

- Легко відслідковувати
- Блокування зі сторони брандмауерів та маршрутизаторів.

Застосовується як швидкий спосіб дізнатися чи порт активний чи закритий, але рідко використовується, так як легко прослідковується і блокується.

- Напіввідкрите TCP сканування– надсилання SYN запиту до порту сервера і чекання SYN-ACK від цілі та не завершує процес рукостискання.

  Переваги:

  - Швидкість, завдяки неповному з'єднанню.
  - Складність у відслідкуванні.

  Недоліки:

  - Основним недоліком є те, що для цього типу сканування потрібно налаштувати IP-пакети, які вимагають спеціальних привілеїв користувача, і це стосується майже всіх операційних систем.

Застосовується для отримання неповної інформації про сервіс, який використовує цей порт, також менш помітний для брандмауера.

- Повне TCP сканування[12], таке ж як напіввідкрите, але уже із завершеним процесом рукостискання.

  Переваги:

  - Повна інформація про сервіс
  - Використання сервісу

  Недоліки:

  - Повільне, так як треба робити додаткові запити на закінчення процесу TCP з'єднання
  - Легко відслідковуване з'єднання

Застосовується для отримання додаткових даних процесу і його застосування.

- UDP сканування[12] – відправлення пакету зі специфічними даними або пустим до UDP портів серверу, якщо порт закритий, то відповідно прийде відповідь, а якщо відповідь не прийшла, тоді порт відкритий або фільтрується. Так як деякі сервіси використовують замість TCP портів UDP, то таке сканування для портів залишається унікальним. Тому судити про переваги та недоліки не має сенсу. Застосовується для сканування UDP портів.
- XMAS сканування – відправлення пакетів з усіма прапорами і навіть FIN прапором, без очікування відповіді від порту. Тобто якщо порт буде відкрито або він фільтрується, то відповідь не прийде, в іншому випадку порт закритий. Метод схожий на UDP сканування, тільки с TCP портами.

  Переваги:

  - Висока складність відслідковування

  Недоліки:

  - Мало інформації про сервіс

Застосовується для тихого сканування відкритих закритих портів. При цьому тестуванні сканер не буде замічений.

Кожне з цих сканувань унікальне і потрібне в різних ситуаціях, тому важко сказати, яке з сканувань найкраще, так як системи для знаходження вразливостей серверу використовує всі сканування, комбінуючи їх для тої чи іншої операції.

2.2.5 Аналіз сервісів

На даному етапі знаходження вразливостей, сервіси тестуються на можливі вразливості, які приведені в базах даних вразливостей. Таких баз даних багато, тому отримання вразливості не стає такою важкою справою, а ось її потрібність і використання, то тут настає важким момент, коли з великої кількості інформації про вразливість того чи іншого сервісу треба вибрати ту, яка допоможе проникнути в систему, чи отримати дані с бази, чи його порушити. Варіативність залежить від хакеру і яку тактику він вибере. Тихо зайти на сервіс

чи положити сервіс для отримання доступу до іншого. Сервісів багато і зазвичай вони знаходяться на стандартних для них портів. Декілька основних сервісів і портів буде наведено в табл. 2.1 взятої з веб-ресурсу[13].

Таблиця 2.1 Стандартні сервіси і їх порти

Порт	Сервіс	Призначення
20	File Transfer Protocol (FTP)	Передача даних
21	File Transfer Protocol (FTP)	Управління командами
22	Secure Shell (SSH)	Доступ до віддаленого серверу
23	Telnet	Доступу до віддаленого серверу, тільки запити будуть не зашифрованими
25	Simple Mail Transfer Protocol (SMTP)	Передача email листів
53	Domain Name System (DNS)	Підміни домен ім’я на IP адресу
80	Hypertext Transfer Protocol (HTTP)	Використання сервісу в інтернеті
110	Post Office Protocol (POP3)	Отримання email листів
119	Network News Transfer (NNTP)	Передачі новин, а саме статей
123	Network Time Protocol (NTP)	Синхронізація часу в мережі

Табл 2.1 Стандартні сервіси і їх порти

Порт	Сервіс	Призначення
143	Internet Message Access Protocol (IMAP)	Передача email листів через TCP/IP
161	Simple Network Managment Protocol (SNMP)	Організація роботи пристроїв в мережі
194	Internet Relay Chat (IRC)	Обмін текстовими повідомленнями клієнт/сервером

Знаючи на якому порту налаштований сервіс можна дізнатися більше інформації про цей сервіс. Його версію і завантаженість. Назва сервісу і його версія вже дає повід для пошуку вразливостей в базі даних по назві та версії, але відібрати їх буде наступним етапом.

2.2.6 Аналіз вразливостей та їх оцінка

Зібравши купу вразливостей з бази даних треба дати їм оцінку для кращої експлуатації ураження і вибрати тактику відповідно до сервісу і типу його пробілу в безпеці, відштовхуючись від цього буде прийнято рішення про подальший план злому. Тому слід розділити надати вразливостям критерії, щоб відповідно до них робити оцінку і для об’єктивного використання ураження. Виділимо наступні критерії:

- Тип вразливості, знаючи його можливо спланувати подальший сценарій розвитку для доступу в систему або отримання додаткової інформації для використання в майбутньому. І різновидів типів дуже багато, наприклад переповнення пам’яті чи її пошкодження, або запуск свого коду на системній машині і т.д.
- Отримання доступу до системи, мабуть це один із найважливіших пунктів, тому що якщо існує вразливість за допомогою якої ти отримаєш

мінімальні права і проникнеш в систему, то знайдене ураження буде критичне і його треба виправляти чи блокувати сервіс, поки зловмисники не експлуатували цей пробіл швидше. Дане порушення може привести до великих збитків.

- Тип доступу до сервісу. Даний пункт набуває таких критеріїв:
  - Віддалений. Сервер знаходиться не в вашій мережі і вам треба підключатися для доступу до нього.
  - Локальний. Сервер знаходиться в вашій мережі LAN і відповідно у вас прямий доступ до нього.
- Складність в експлуатації ураження. Розділимо складність в реалізації вразливості на декілька порогів:
  - Низький, дана складність використання пробілу в безпеці передбачає легку експлуатацію даного ураження і потребує від зловмисника не високого рівня знань.
  - Середній, на даному порозі хакеру потрібно буде проаналізувати вразливість і відповідно до своєї системи, використати її. Тут треба писати код з нуля або дописувати його і це потребує додаткових знань від зловмисника для реалізації плану.
  - Високий, потребує і високого рівня знань від людини, яка буде аналізувати, використовувати дірку в системі. Як правило дані пробіли в безпеці не завжди реалізуються через їхню складність, але якщо це коштує того, то треба експлуатувати.
- Аутентифікація, тобто її уникнення. Якщо на сервісі існує аутентифікація і є можливість обійти її, то це один із варіантів доступу до системи, якщо у сервісу будуть привілеї для його використання.

2.3 Парсинг інформації

Парсинг[14] – синтаксичний аналіз даних, при якому один тип даних перетворюється в інший. В нашому випадку інформація з веб-ресурсів, а саме

необроблені дані HTML, перетворюються в організовану структуру даних, в якій можливе подальше маніпулювання інформації після аналітичної обробки даних, по певному критерію. Критерій задається відповідно до веб-ресурсу і знаходження на ньому потрібного типу даних. Критерії виражаються HTML елементами:

- <p> – елемент, який відповідає за параграф.
- – елемент довільного ділення даних, для її подальшої графічного розділення.
- <a> – елемент, який відповідає за посилання.
- <h1> ... <h4> – елемент, який відповідає за заголовок.
- , <ul>, <ol> – елементи, які відповідають за відображення списків.
- – елемент, для відокремлення частини тексту.
- Href – атрибут для елементу <a>, який приймає як параметр посилання.
- Text – атрибут елементу, який відображає текст і несе корисну інформацію, яка може бути корисною.
- Class – атрибут елементу HTML, який класифікує елемент під стиль, який задається як параметр.

Знаючи HTML розмітку веб-ресурсу, то за допомогою комбінування критеріїв, можливо отримати дані з сайту використовуючи при цьому інструменти програмування, що забезпечать собою ефективне використання даного способу вибору інформації. На рис 2.3 зображено приклад HTML розмітки сайту www.kpi.ua.

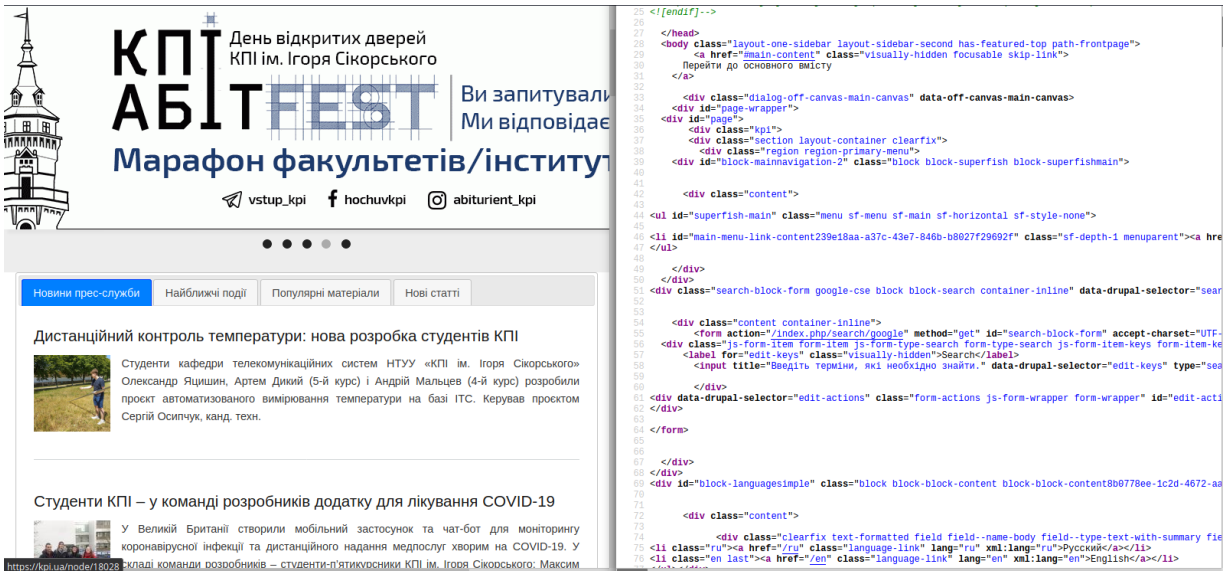


Рисунок 2.3 − HTML розмітка сайту www.kpi.ua

Використовуючи критерії з HTML і розташування потрібної інформації в розмітці, то отримання інформації полегшується, шляхом організованої структуризації даних.

2.4 Визначення вимог і завдання

2.4.1 Загальний функціонал системи знаходження вразливостей

Система пошуку вразливостей серверу – це програма, яка виявляє та тестує сервер на наявність уражень в системі, аналізує виявлені пробіли в безпеці та дає їх оцінку.

Система пошуку вразливостей повинна виконувати наступні функції:

- Сканування мережі на наявність уражень
- Аналіз вразливостей, відповідно до їх степені ураження
- Загальний рапорт про безпеку серверу

Загальною цілю даної програми – повідомити користувача про можливі ураження в системі, які можуть привести до краху його сервісу.

2.4.2 Визначення завдань

В даній системі можна виділити основні завдання:

- Сканування локальної мережі на наявність серверів.

- Сканування портів знайдених серверів.
- Визначення сервісу, який стоїть на порту.
- Пошук вразливостей по веб версіям бази даних.
- Оцінка ураження.
- Перехід на першоджерело вразливості.

ВИСНОВОК РОЗДІЛУ 2

В даному розділі розглянута інформація про пошук вразливостей та де можна віднайти інформацію для їх аналізу. Було проаналізовано такі веб-ресурси:

- CVEDETAILS
- EXPLOIT-DB

Де можливо віднайти інформацію відповідно до продукту та його версії, чим можемо в подальшому скористатися. Також, наявність в EXPLOIT-DB інформації про про етапи експлуатування ураження і його коду, робить перевірку більш розгорнутою, тобто отримана інформація, допоможе скоректувати оцінку вразливості.

Було виділено основні етапи для пошуку вразливостей:

- Знаходження мереж, які в подальшому будуть атаковані.
- Сканування портів серверу, на наявність сервісів, які можуть бути атаковані.
- Аналіз сервісів, які знайшли на попередньому етапі на наявність вразливостей.
- Аналіз вразливостей і їх оцінка на можливе проникнення в систему чи порушення його роботи.

Пояснено чому важливий кожний із етапів і його значення для системи в цілому.

Сформовано основні вимоги до функціоналу системи, а також поставлено задачі для їх реалізації.

РОЗДІЛ 3

РОЗРОБКА СИСТЕМИ

3.1 Вибір технологій та засобів проектування

3.1.1 Вибір платформи

Більшість серверів розташована на Unix подібних системах, тому що сама оболонка даної системи зручна в використані. Представлена організація ОС дає змогу розділення користувачів на окремі процеси і надавання їм прав для використання системи, що робить їх безпечнішими. Але є і сервера на Windows ОС, якщо там налаштовано мало сервісів і не треба організації, то для користувача, який має малий досвід у програмуванні Windows стає найкращим варіантом. Потужність сервісів і серверів росте, тому і варіант з використанням Windows ОС, як платформу для системи не розглядається. Декілька переваг Unix над Windows описані на веб-ресурсі[15] в нашому випадку:

- Unix ОС безкоштовна, в той час як треба платити за Windows і сервісів, які буду там налаштовані.
- Unix сервера можна модифікувати, коли вони запущені, без рестарту серверу, що не скажеш про Windows.
- Unix більш захищена система, тому що безпека сервісів на ній більша, а Windows системи працюють на відомих сервісах, що прославляються своєю вразливістю.

Досить глянути на статистику, яку представив відомий зарубіжний хостинговий сервіс vps.net[16] по ОС на серверах рис. 3.1. Як видно зі статистики біля 90 % користувачів сервісу використовують Unix подібні операційні системи.

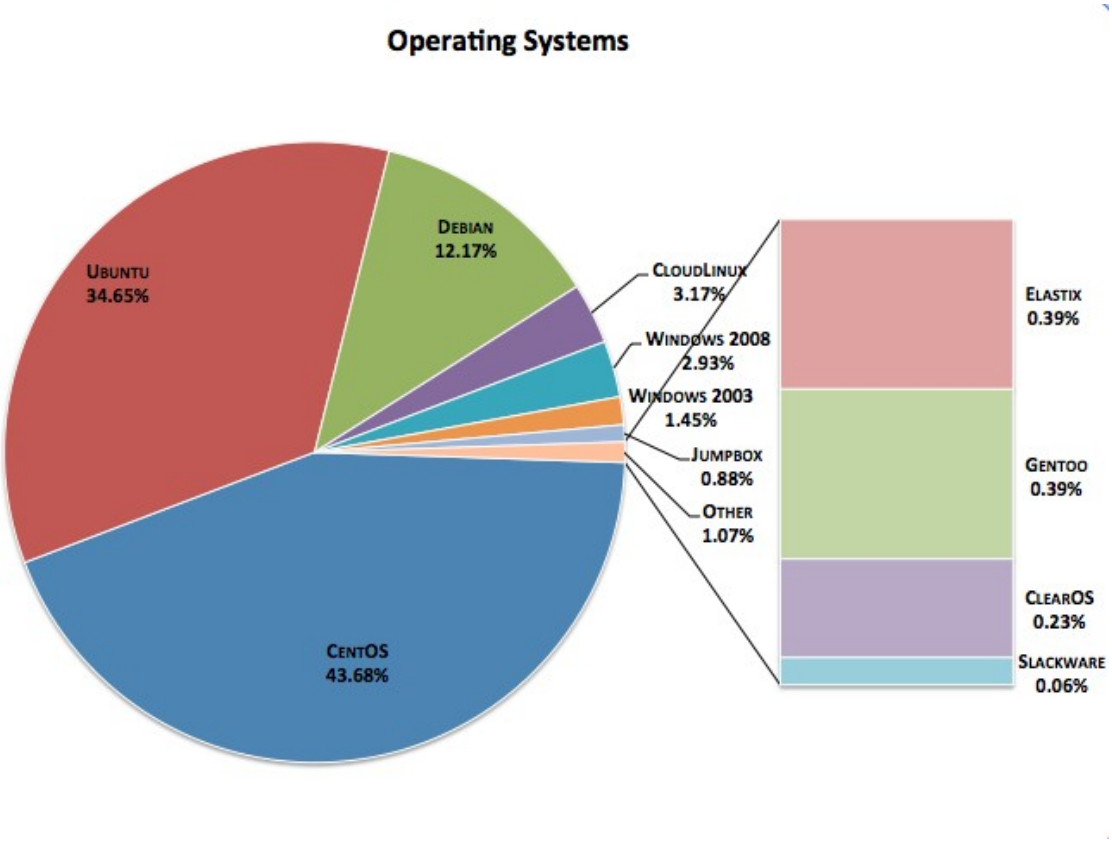


Рисунок 3.1 − Статистика використання ОС на серверах

Виходячи з вище перерахованих фактів наша система буде розроблятися на базі Unix подібної системи для легкої імплементації її в сервер.

3.1.2 Вибір мови програмування

Для виконання поставлених задач нам потрібна мова програмування, яка буде мати наступні характеристики:

- Наявність ООП для реалізації структур вразливостей
- Можливість або наявність бібліотеки для сканування мережі
- Підтримка API для EXPLOIT-DB
- Можливість парсингу веб інформації
- Створення графічного інтерфейсу
- Легкість в імплементації на сервер

По даним вимогам було вибрано мову програмування Python[17], тому що вона легка в освоєні і має багато вже написаних модулів, для реалізації наших задач. Важливу роль у виборі відіграла наявність модулю для синтаксичного аналізу

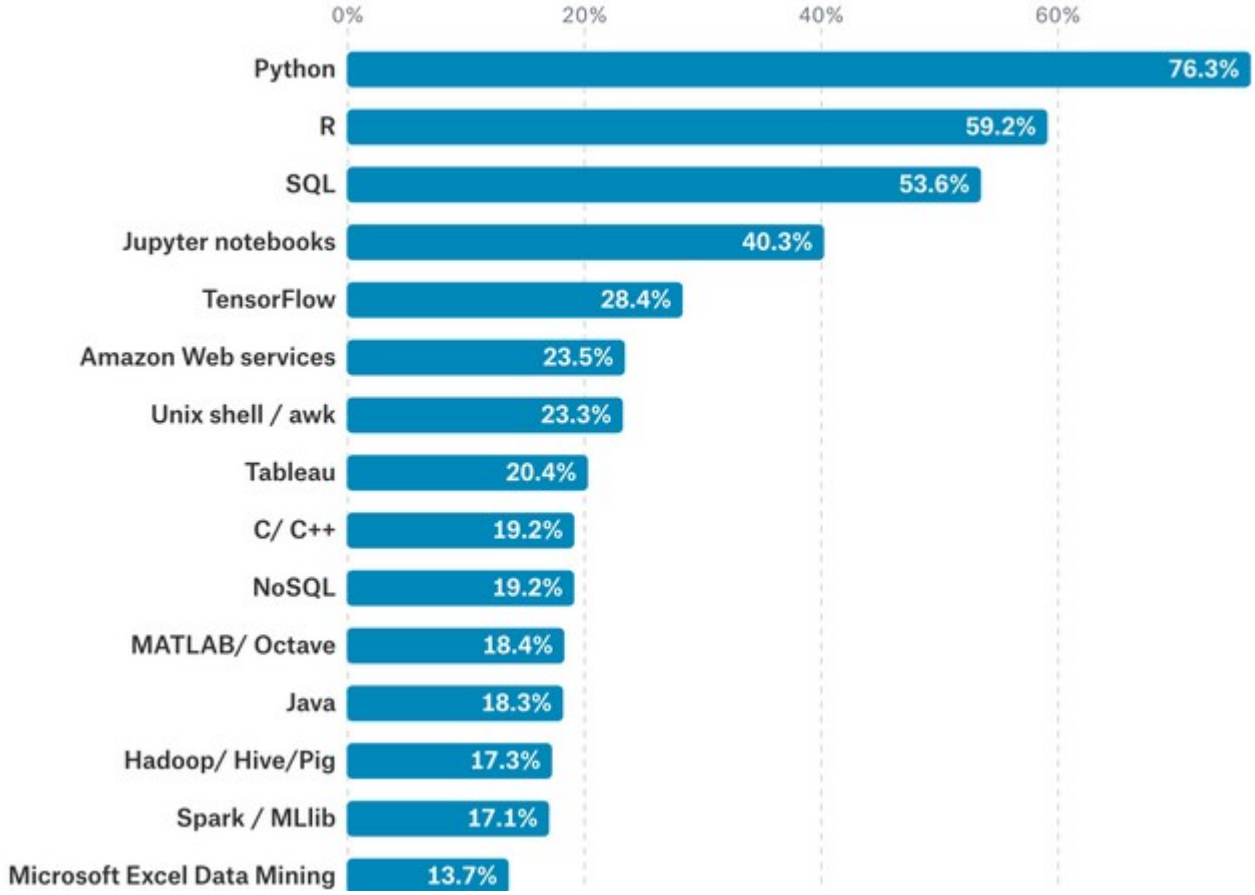


Рисунок 3.2 − Діаграма використання мови програмування для аналізу інформації.

інформації з веб-ресурсу і можливість реалізації інтерфейсу на інші платформи. Ці критерії роблять вибір мови програмування Python, це більш привабливими. Беручи до уваги високу популярність Python, як мови, яка використовується для аналізу даних рис. 3.2.

Беручи до уваги опитування спеціалістів по маніпулювання даних, яке було проведене в 2017 році[18]. Їм поставили запитання, яку б мову програмування ви б вибрали для маніпуляції інформації, то 8 з 10 людей на перше місце поставили мову програмування Python. Це ще одна з вагомих причин в сторону вибору цієї мови програмування.

3.1.3 Вибір бібліотек

Для реалізації даної системи, було прийнято рішення про використання таких бібліотек:

- Kivy, графічний інтерфейс.
- Python-nmap, сканування мереж.

- Request, запити до сервера.
- Socket, підключення до серверу.
- Urllib3, отримання інформації від веб-ресурсу.
- BeatifulSoup, парсинг веб-ресурсів.
- Webbrowser, відкриття вікна браузера

Розглянемо кожну з бібліотек, тобто їх короткий опис.

3.1.4 Kivy

Kivy[19] – безкоштовний фреймворк для реалізації графічного інтерфейсу. В його наявності є велика кількість віджетів за допомогою, яких можна зробити інтерактивність в програмі. Є доступ до використання комп’ютерної миші і клавіатури. Kivy підтримується на Unix подібних системах, використовуючи OPENGL ES 2, як затверджують розробники, то деяка кількість функцій реалізовані на Cython, тобто на C. OPENGL ES 2 і так досить швидкий і на додачу до цього важкі функції реалізовані на C. Розглянемо приклад простого коду Kivy фреймворка і його графічний вивід.

Приклад коду Kivy:

```
from kivy.app import App
from kivy.uix.button import Button
class DiplomExample(App):
    def build(self):
        return Button(text='DiplomExample')
DiplomExample().run()
```

Рисунок 3.3 − Графічний вивід прикладу Kivy

Як видно з прикладу на рис. 3.3 за досить невеликих зусиль було створено графічне вікно. Великим плюсом Kivy в тому що він має велику і зручну документацію по його використанню і вона інтуїтивно зрозуміла. Для того щоб зрозуміти механізм життя Kivy програми, роздивимося рис.3.4, яким пояснюють самі розробники цикл життя Kivy програми[19].

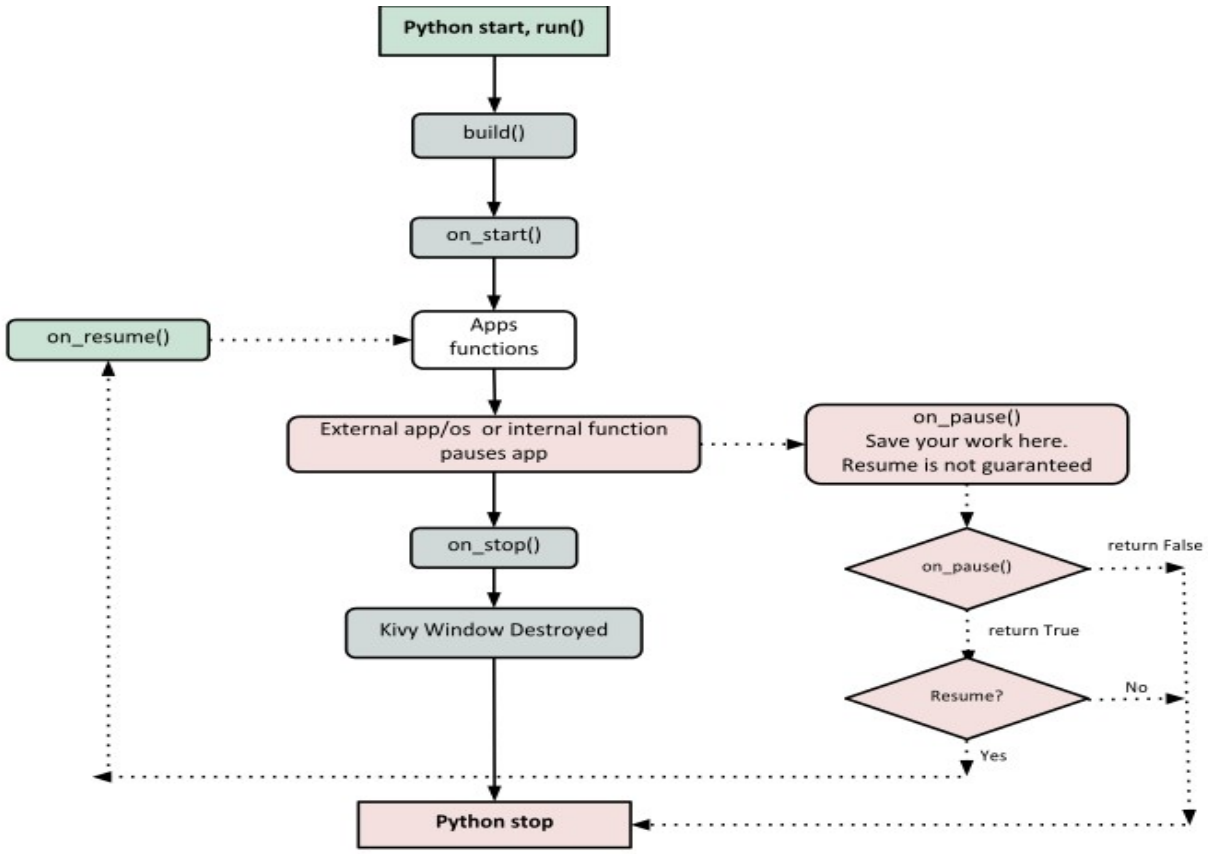


Рисунок 3.4 − Цикл життя Kivy програми

Для того щоб не нагромаджувати логічний код графічним дизайном, в Kivy є можливість відділити вибір кольору, шрифту, розмірів від основного файлу програми до допоміжного .kv формату, що підвищує читабельність такого коду.

Приклад коду .kv файлу:

Button:

text : “Diplom Example”

size: 10 20

color: 0.13, 0.16, 0.19, 1

Рисунок 3.5 − Приклад графічних змін при внесені .kv файлу

Як видно з рис.3.5 текст, колір і розміри кнопки змінились.

3.1.5 Python-nmap

Python-nmap[20] – безкоштовний фреймворк, який допомагає просканувати мережу і отриманий результат проаналізувати на наявність сервісів. Python-nmap є аналог системної утиліти nmap, яка є популярним сканером мережі в наші дні. Виділено її основні переваги:

- Безкоштовна, що дає їй велику популярність.
- Підтримка різних типів запитів TCP і UDP
- Підтримка Unix систем
- Детальна документація

3.1.6 Socket. Request. BeautifulSoup. Urllib3. Webbrowser.

Socket[21] – бібліотека для створення підключення типу клієнт сервер.

Request – python бібліотека, яка надсилає запити і зберігає відповідь від серверу. Для вебсайту ця відповідь буде у вигляді html формату.

BeautifulSoup[22] – інструмент для трансформації html сторінки в організовану інформацію, в якій можна з легкістю віднайти потрібну інформацію.

Urllib3[23] – бібліотека схожа на request, тобто виконують надсилання запиту і його збереження, тільки формат збереження відрізняється, що збільшує наш арсенал запитів відповідно до ресурсів.

В купі вище перераховані бібліотеки дають змогу парсити інформацію з інтернету в потрібному форматі і з потрібних серверів. Тобто за допомогою socket встановлюється підключення, потім використовуючи request або urllib3 надсилаємо запити до вебсайтів і отримуємо від них відповідь, за допомогою beautifulsoup парсимо надану інформацію і отримуємо організовані дані від веб-ресурсу.

Webbrowser – відкриває вікно інтернет браузера.

3.1.7 BeautifulSoup.

В якості основного методу парсингу інформації, тобто синтаксичного оброблення даних, вибір пав на бібліотеку BeautifulSoup, яка перетворює HTML розмітку в організовану структуру даних, в якій за допомогою HTML елементів можливо отримати потрібну інформацію. Потрібні функції з модуля:

- BeautifulSoup(text) – метод, який оброблює текст HTML або XML з звичайних символів ASCII або UTF-8 в Unicode організовану структуру даних, яка підлягається подальшої обробки. Функція виконує роль синтаксичного аналізу тексту з розбиттям його на категорії відповідно до HTML елементів. Після перетворення тексту, доступ до елементів HTML відбувається шляхом пропису шляху до цього елементу. Приклад доступу до всіх заголовків h1:

  soup = BeautifulSoup(text) soup.h1

- find_all(element) – метод, який знаходить всі елементи, які відповідають певним критеріям, що надає можливість отримання доступу до атрибутів цих елементів для подальшого маніпулювання даними. Метод використовується для полегшення пошуку потрібної інформації у організованій структурі, яка була сформована завдяки виклику функції BeautifulSoup(text). Приклад знаходження всіх елементів з наявним атрибутом класу “srrowns”:

  soup = BeautifulSoup(text)

  div = soup.find_all("div", {"class" : "srrowns"})

- text() – метод, який повертає текст з елемента уже готової організованої структури BeautifulSoup.

3.2 Основні рішення з реалізації системи

3.2.1 Конфігурація віртуального середовища

Для легшої імплементації системи, було прийнято рішення створити віртуальне середовище[24] з потрібною версією Python і її бібліотеками. Для цього використано вбудовану функцію Python venv. Приклад створення:

python3 -m venv шлях_до_середовища/назва_віртуального_середовища

Створене віртуальне середовище буде містити мінімум пакетів для функціональної роботи Python3, на відміну від повної версії, яка складається з тисячі файлів, тому більш ефективніше використання віртуального середовища з обмеженою кількістю бібліотек. Причини застосування віртуального середовища:

- Швидкість створення, так як частина функціоналу береться з системної версії Python для роботи такого середовища лиш потрібно інсталювати потрібні для роботи пакети в середовище.
- Надійність системи, так як програма використовую бібліотеки строго визначених версій, то будь-яке порушення при інсталяції пакетів для

роботи системи може призвести до частинної відмови роботи або взагалі до повного припинення.

- Вирішення проблеми конфліктів з системними пакетами, створення окремого незалежного середовища для функціонування конкретного процесу вирішує проблему впливу на це середовище зі сторони інших модулів.

Рисунок 3.6 − Файлова структура віртуального середовища

На рисунку 3.6 можна переглянути приблизну структуру віртуального середовища та її компоненти. Кожний користувач загрузивши програму, не буде турбуватися з приводу встановлення додаткових модулів до програми, адже там буде наявне середовище, в якому всі модулі присутні, спрощуючи запуск

програми на машині клієнта. Якщо б користувач не зміг би встановити додаткові бібліотеки чи потрібну версію Python, то і система не буде працювати.

3.2.2 Огляд класів системи

Python, як мова програмування підтримує, як ООП так і функціональне програмування, тому для графічного інтерфейсу Kivy було створено систему класів, більшість з яких належать Kivy фреймворку, а для реалізації пошуку і аналізу вразливостей використано функціональне програмування.

З Додатку 1 можна побачити залежність класів, яка реалізована в даній програмі. Класи, які потрібні для інтерфейсу:

- App – клас Kivy фреймворку для запуску циклу життя програми, до того часу поки не прийде повідомлення про закриття або не виникне помилка в програмі, яка призведе до закриття.

  Основний метод:

  - run() – запуск циклу програми.

- FloatLayout – клас віджету Kivy, який дозволяє розташувати виділену область в будь-якій частині програми графічного інтерфейсу. Накладати ці області чи створити дочірні і відповідно до позицій буде залежати загальне зображення інтерфейсу програми.

  Основні методи:

  - add_widget(widget) – додати віджет в область
  - remove_widget(widget) – видалити віджет з області

- GridLayout – клас віджету Kivy, в якому можливо розбити область на колонки і стовпчики у виді таблиці для зручного і логічного зображення в них інформації. Так як і з FloatLayout можна створювати дочірні віджети, які будуть відображатися у вигляді таблиці.

  Основні методи:

- add_widget(widget) – додати віджет в область
- remove_widget(widget) – видалити віджет з області

- ButtonBehavior – клас Kivy, який надає будь-якому віджету функції кнопки, Основний метод:
  - on_press() – виконання дії, коли кнопка буде нажата.
- ButtonGrid – клас створений для реалізації таблиці з кнопок, тому він наслідує клас ButtonBehavior і GridLayout, що дає змогу створити інтерфейс з інтерактивним відкликом.
- VulnFind – клас створений для об’єднання функцій знаходження і аналізу вразливостей з графічним інтерфейсом. Для отримання даних з інтерфейсу і передачу її в методи для сканування або пошуку, які в свою чергу повернуть дані і вони ж виведуться на графічний інтерфейс. Алгоритм пошуку вразливостей показаний в Додатку 2.

  Основні методи:
  - scan(ip, port) – функція, яка сканує мережу по вибраному діапазону ip адрес, якщо знаходить, то починає сканування по вибраному діапазону портів. Функція повертає python словник, в якому ключ є портом, а значенням сервіс і його версія.
  - find(service) – функція, виконує пошук по базам даних або їх веб версіям, в залежності від того доступна база чи ні. Для отримання інформації з веб-ресурсів інформація на них розбирається і організовується, потім вибирається інформація про вразливості і їм надається середня оцінка, вони сортуються відповідно оцінці і повертаються в графічний інтерфейс.
  - thread_scan(ip, port, num) – аналогічна функція до scan, тільки в цьому випадку сканування розділяється на num кількість ядер

- для пошуку ip,а потім аналогічно і сканування портів. Реалізовується шляхом надання виклику scan на ядрах.
- thread_find(service, num) – аналогічна функція до find, тільки в цьому випадку сканування виконується на num кількості ядер, для пошуку вразливостей. Реалізовується шляхом надання виклику find на ядрах.

Для опису властивостей уражень створено клас Vuln і йому надані наступні атрибути:

- Name. Назву.
- Cve. Номер унікального CVE.
- Cve_url. Посилання на CVE у веб-ресурсі[9].
- Type. Тип вразливості.
- Verf. Перевірене ураження чи ні.
- Urls. Посилання на джерела р можливою додатковою інформацією.
- Exploit. Посилання на план використання вразливості.
- Score. Оцінка вразливості.

В даному класі, немає методів, адже він був створений лише для опису характеристик, які потрібні користувачу, для більшого розуміння в потрібності ураження.

Розглянемо по окремості методи, які реалізовані в scan:

- parse_kivy(ip, port_range) – функція, в якій розбивається IP адреси і порти до формату виклику функції сканування в nmap. Також в даній функції перевіряється на правильність введених даних користувачем, а саме IP та діапазон портів.
- nmap.scan – метод бібліотеки nmap, яки виконує задані в параметрах запити по IP адресам, портам і зберігає інформацію.

- look_up_ports(nmap, ips) – функція, яка перебирає знайдені сервера і зберігає тільки назви сервісів в словник у виді { port : name}, порт ключ і назва сервісу разом з версією.

Розглянемо по окремості методи, які реалізовані в методі find:

- exploit_db(url, product, version) – метод, який шляхом парсингу знаходить по product і version вразливість на веб-ресурсі[10] Зберігаючи всю потрібну інформацію в клас Vuln, а саме ім’я, тип вразливості, перевірена чи не перевірена, посилання на використання вразливості, якщо є.
- cve_details(urk, product, version) – метод, який шляхом розбору інформації знаходить вразливості по product і version на веб-ресурсі[9]. Зберігаючи потрібну інформацію, а саме ім’я, cve ідентифікатор, cve_url посилання на вразливість, тип вразливості, перевірене чи не перевірене ураження, оцінка.
- scrap_vuln_info(product, version) – метод, який виділяє можливі веб-ресурси для знаходження вразливостей, виклики функцій exploit_db і cve_details.

3.3 Реалізація графічного інтерфейсу

Для реалізації графічного інтерфейсу було вибрано python бібліотеку, щоб не нагромаджувати логічний код стилістичними функціями і винести його в окремий файл. Тому в даному пункті розберемо дизайн системи і як він реалізований в Kivy, буде розглянуто синтаксис .kv файлу. Можливості користувача описані в Додатку З.

Для початку користувач має ввести діапазон Ip адрес, портів і натиснути на кнопку, яка починає сканування портів. Для цього потрібно 2 поля введення і кнопка рис. 3.7, яка б повідомила про наявність даних для їх подальшої обробки.

Приклад коду стилізації.

```
TextInput:
    id: target_scan
    hint_text: "127.0.0.1"
    size_hint: 0.18, 0.05
    pos_hint: {"center_x" : 0.35, "center_y" : 0.9}

Label:
    font_size : 20
    text : ":"
    pos_hint: {"center_x" : 0.455, "center_y" : 0.9}

TextInput:
    id: port_range
    hint_text: "1-65535"
    size_hint: 0.1, 0.05
    pos_hint: {"center_x" : 0.52, "center_y" : 0.9}
Button:
    id: btn_scan
    text: "Scan"
    size_hint: 0.18, 0.06
    on_press: root.scan(btn_scan)
    pos_hint: {"center_x" : 0.7, "center_y" : 0.9}
```

Рисунок 3.7 − інтерфейс введення IP адреси і діапазону портів

Розглянемо за що кожний із специфікаторів відповідає:

- Label – віджет, в якому розміщається текст, який було задано в параметрах для зображення.
- TextInput – віджет введення тексту, який буде оброблятися програмою при натисканні на кнопку.
- Button – віджет кнопки, який при натисканні буде викликати функцію, що задано в параметрах.

Розглянемо за що кожний із параметрів відповідає:

- id – присвоює даному віджету ідентифікатор, за допомогою якого в програмній частині можна взаємодіяти з ним, наприклад зчитати інформацію.
- hint_text – прихований текст, який виконую роль прикладу введення даних і якщо текст, не буде введений, то програма зчитає hint_text.
- size_hint – розмір віджету відносно ширини і висоти батьківського віджету, вданому випадку вікна програми. Якщо ми змінимо розмір вікна, то і відповідно розмір віджету зміниться.
- pos_hint – розташування віджету, відносно ширини і висоти батьківського віджету, тобто вікна програми.
- Text – текст, який буде виведений на віджет.
- font_size – розмір шрифту тексту.
- on_press(func) – виклик функції func при натискні на віджет.

Після обробки запиту користувача, треба вивести інформацію в зручному форматі, так як функція повертає python словник, де ключ це порт, а значення це список з таких параметрів сервісу:

- State. Його стан:
  - Open. Відкритий.
  - Filtered. Фільтрований.
- Name. Скорочене ім’я сервісу.
- Product. Повне ім’я сервісу.
- Version. Версія продукту.

Для виведення даної інформації, було вирішено виводити її у форматі таблиці з наведеними вище полями. Для створення такої таблиці, в Kivy реалізований клас GridLayout, який дозволяє розташувати інформацію у вигляді таблиці рис. 3.8. Розглянемо приклад коду для виведення такої інформації:

```
GridLayout:
    id: port_top
    cols: 5
    padding: 50, 0
    size_hint_x: 1.11
    pos_hint: {"top" : 1.4}
    Label:
        text: 'Port'
        color: 0.13, 0.16, 0.19, 1
    Label:
        text: 'State'
```

```
        color: 0.13, 0.16, 0.19, 1
    Label:
        text: 'Name'
        color: 0.13, 0.16, 0.19, 1
    Label:
        text: 'Product'
        color: 0.13, 0.16, 0.19, 1
    Label:
        text: 'Version'
        color: 0.13, 0.16, 0.19, 1
```

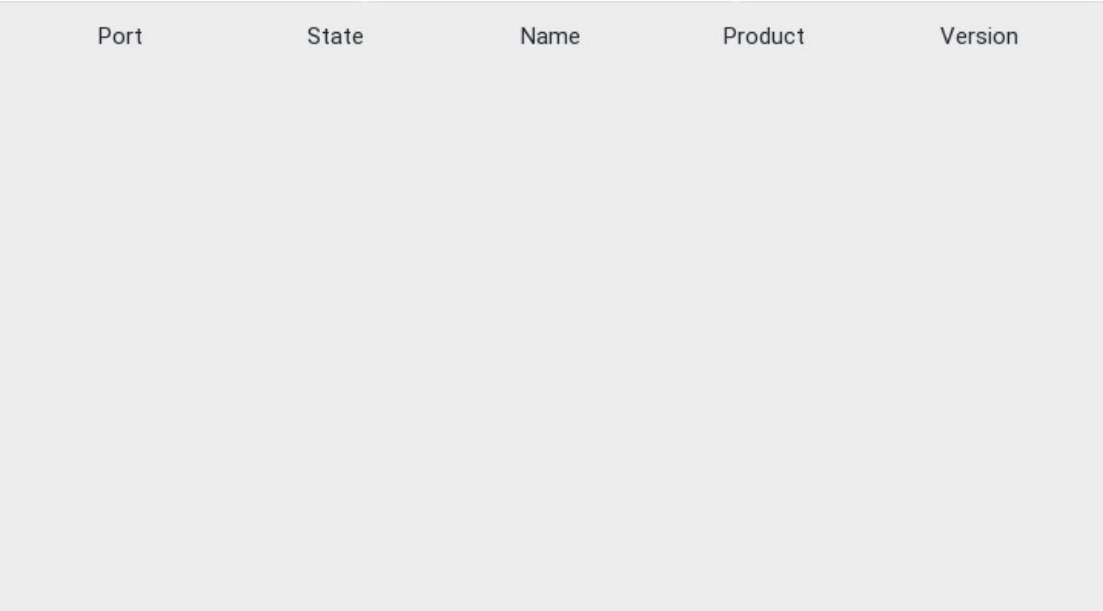


Рисунок 3.8 − Приклад інтерфейсу таблиці для виведення інформації

В GridLayout добавлені нові параметри віджету, а саме:

- cols – параметр кількості колонок, які будуть в нашій таблиці.
- Padding – внутрішній відступ віджету від його країв.
- Color – колір тексту віджету.

Зазначмо, що в файлі .kv формату визначено кількість колонок, але не рядів, бо до запиту користувача на сканування серверу невідомо скільки сервісів знайде програма, тому визначення кількості рядів буде вирішуватися програмно після обробки запиту користувача. Для виведення інформації після сканування в таблицю, вирішено розробити клас ButtonGrid, який наслідує класи ButtonBehavior і GridLayout, для реалізації функції натискання на поле для його вибору рис 3.9. Приклад .kv коду:

```
GridLayout:
    id: grid_scan
    cols: 5
    padding: 40, 0
    top: self.height
    size_hint_x: 1.11
    height: self.minimum_height
    row_default_height: '35dp'
    row_force_default: True
```

Port	State	Name	Product	Version
22	open	ssh	OpenSSH	7.6p1 Ubuntu 4ubuntu0.3
111	open	rpcbind		2-4
631	open	ipp	CUPS	2.2
2049	open	nfs_acl		3

Рисунок 3.9 − Приклад інтерфейсу таблиці із заповненими даними

В GridLayout добавлені нові параметри віджету, а саме:

- top – відступ зверху для кожного віджету в GridLayout.
- Height – висота віджету.

- row_default_height – стандартна висота поля таблиці.
- row_force_default – зміна розміру поля при зміна розміру таблиці.

Аналогічно і для пошуку вразливостей для сервісу обрана таблиця GridLayout, тільки з іншими назвами колонок і значенням параметрів, а саме:

- Port/Product
- CVE
- Type
- Score
- Description

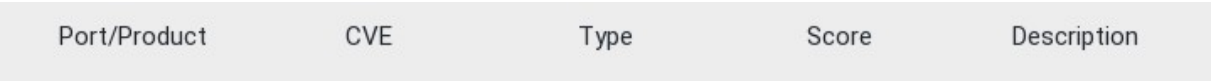


Рисунок 3.10 − Приклад інтерфейсу таблиці для вразливостей

Для зручного користування програмою сканування портів і пошук вразливостей було розділено вкладеннями, також добавлена додаткова вкладка для інформації про систему і її використання. І програма набуває загального вигляду рис. 3.11.

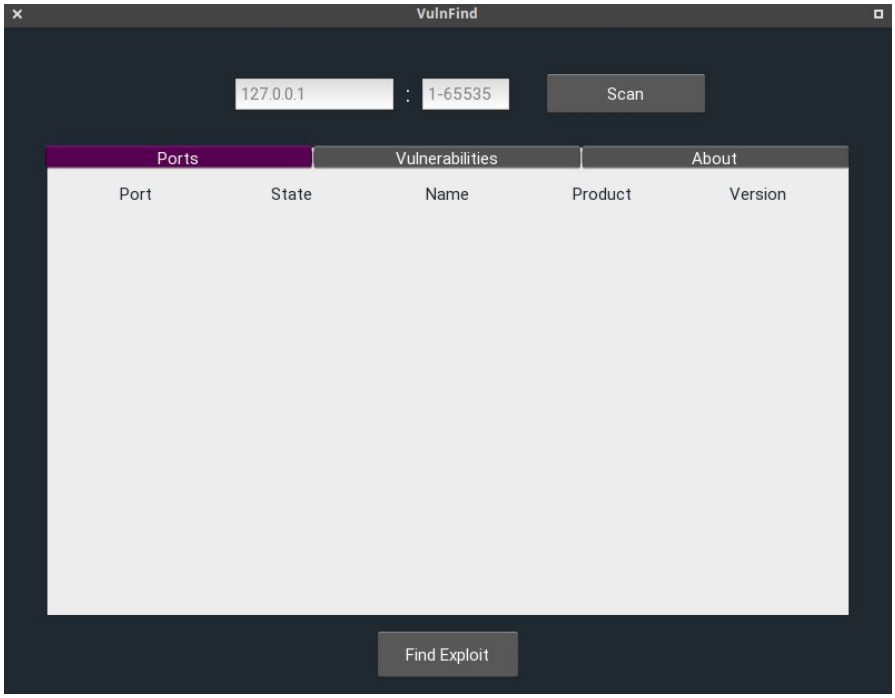


Рисунок 3.11 − Загальний вигляд інтерфейсу програми

ВИСНОВОК ДО РОЗДІЛУ 3

В даному розділі було розглянуто платформу, для якої розробляється система знаходження вразливостей з детальним її аналізом вибору. Було вирішено мову програмування, на якій система буде виконана, а також допоміжні бібліотеки для реалізації інтерфейсу, пошуку по серверу і по веб-ресурсам.

Розглянуто можливість про розробку програми у віртуальному середовищі для полегшення вбудуванню системи на сервер користувача. Також, розглянуто ієрархію класів для забезпечення роботи системи з графічним інтерфейсом. Описані методи для взаємодії системи з графічним інтерфейсом. Детально розібрано дизайн інтерфейсу програми для взаємодії с користувачем.

ВИСНОВКИ

В даному дипломному проекті розглянуто реалізацію системи знаходження вразливостей серверу, що має підвищувати його безпеку.

В ході даної роботи розглянуто поняття вразливості та втрати компаній від ураження ними, також більш детально ознайомилися з важністю наявності системи пошуку вразливостей на сервері. Проаналізовано вже готові варіанти реалізації системи пошуку для виділення переваг і недоліків тої чи іншої системи.

Ознайомлення з процедурою знаходження сервісів на серверів,а також пошуку вразливостей, тобто виділення основних етапів для їх виявлення. Розглянуто такі веб-ресурси як www.cvedetails.com і www.exploit-db.com, звідки можна отримати інформацію шляхом парсингу про вже існуючі вразливості. Розроблено структурну ієрархію класів для забезпечення оптимальної роботи графічного інтерфейсу і методів знаходження вразливостей.

Дана система знаходження вразливостей реалізована під популярну для серверів платформу Unix ОС. Їй доступні функції пошуку по мережі, за допомогою різних типів запитів до серверів, які він може й не вслідкувати. Система є швидко налаштованою через її реалізацію в віртуальному середовищі. Доступні функції знаходження інформації про вразливості шляхом пошуку по конкретних веб-ресурсах. Завдяки реалізованому графічному інтерфейсу використання даної системи стає комфортним.

ПЕРЕЛІК ПОСИЛАНЬ

1. Penetration Testing: A Hands-On Introduction to Hacking by Georgia Weidman (Author) 1st Edition, June 14, 2014 – p. 133-134, p.142-144, p 146-149
2. The world's largest cyberattacks [Електронний ресурс] – Режим доступу: https://outpost24.com/blog/top-10-of-the-world-biggest-cyberattack
3. Why do hackers hack [Електронний ресурс] – Режим доступу: https://www.appknox.com/blog/why-do-hackers-hack
4. Nikto [Електронний ресурс] – Режим доступу: https://cirt.net/nikto2
5. Rapid 7 Nexpose [Електронний ресурс] – Режим доступу: https://www.rapid7.com/products/nexpose/
6. Metasploit [Електронний ресурс] – Режим доступу: https://www.varonis.com/blog/what-is-metasploit/
7. База даних [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85
8. Common Vulnerability Scoring System [Електронний ресурс] – Режим доступу: https://www.first.org/cvss/
9. Cvedetails [Електронний ресурс] – Режим доступу: https://www.cvedetails.com/
10. Exploit-db [Електронний ресурс] – Режим доступу: https://www.exploit-db.com/

11. Hacking: The Art of Exploitation, by Jon Erickson 2nd Edition, February 4, 2008 – p.258-259, p. 264 -268.

12. An Introduction to Computer Networks by Peter L Dordal, Edition 1.9, 2014 – p. 337, p/ 367, p764, p.779

13. Service Name and Transport Protocol Port Number Registr [Електронний ресурс] – Режим доступу:

https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

14. Парсинг [Електронний ресурс] – Режим доступу:

https://ipipe.ru/info/parsing

15. Windows Server vs Linux [Електронний ресурс] – Режим доступу:

https://phoenixnap.com/blog/linux-vs-microsoft-windows-servers/

16. Operation System Deployment Stats [Електронний ресурс] – Режим доступу:

https://www.vps.net/blog/operating-system-deployment-stats/

17. The Python Tutorial [Електронний ресурс] – Режим доступу:

https://docs.python.org/3/tutorial/

18. Key Tools for Data Science [Електронний ресурс] – Режим доступу:

https://mkhalusova.github.io/blog/2018/11/12/data-science-tools

19. Kivy framework [Електронний ресурс] – Режим доступу:

https://kivy.org/doc/stable/api-kivy.html

20. Nmap [Електронний ресурс] – Режим доступу:

https://nmap.org/

21. Socket Programming in Python [Електронний ресурс] – Режим доступу:

https://realpython.com/python-sockets/

22. BeautifulSoup [Електронний ресурс] – Режим доступу:

https://www.crummy.com/software/BeautifulSoup/

23. Extensible library for opening URLs Python [Електронний ресурс] – Режим доступу:

https://docs.python.org/3/library/urllib.request.html

24. Creation of virtual environments [Електронний ресурс] – Режим доступу:

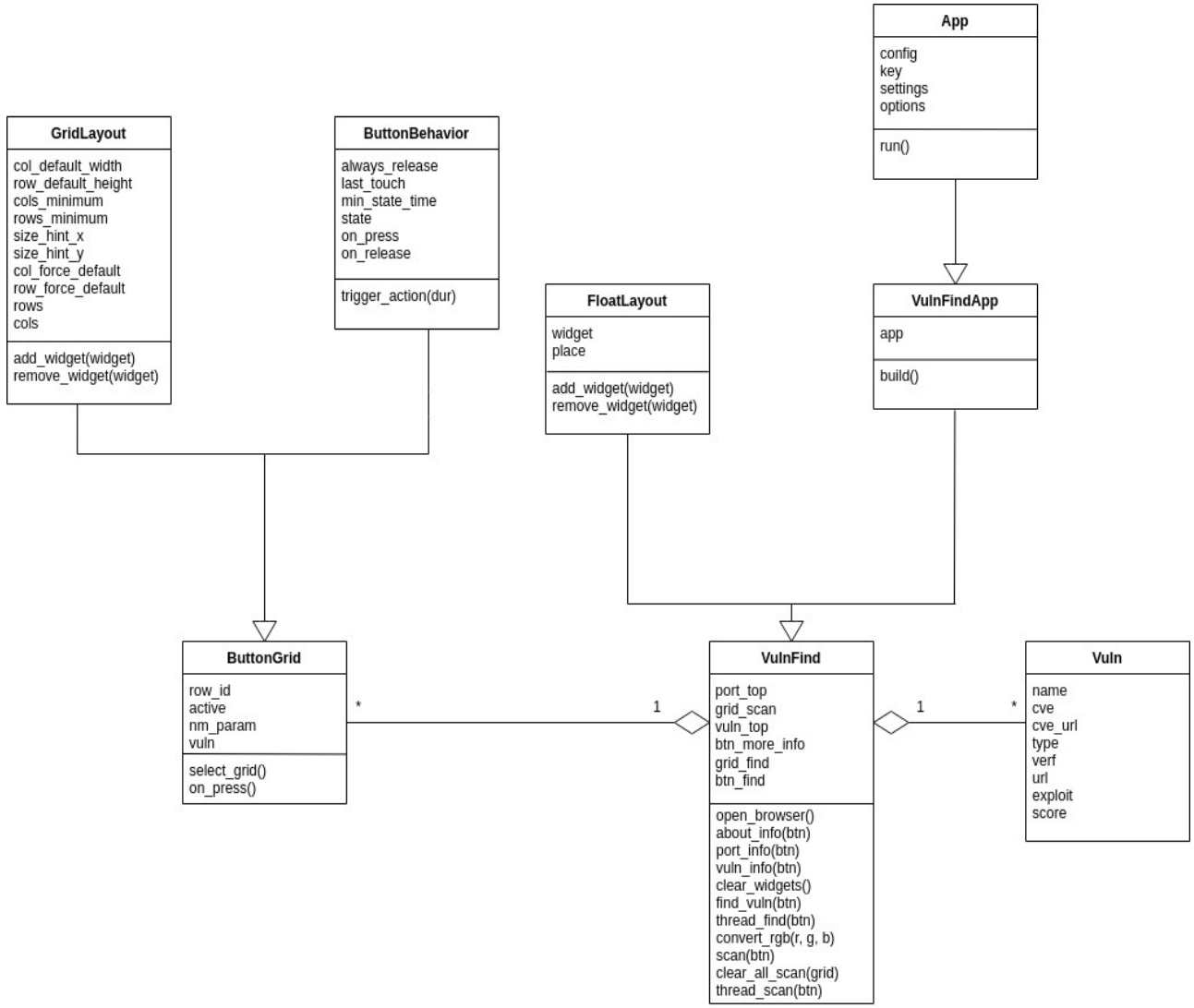
https://docs.python.org/3/library/venv.html

Додаток 1

до дипломного проєкту

на тему: «Система знаходження вразливостей серверу»

Київ - 2020 року



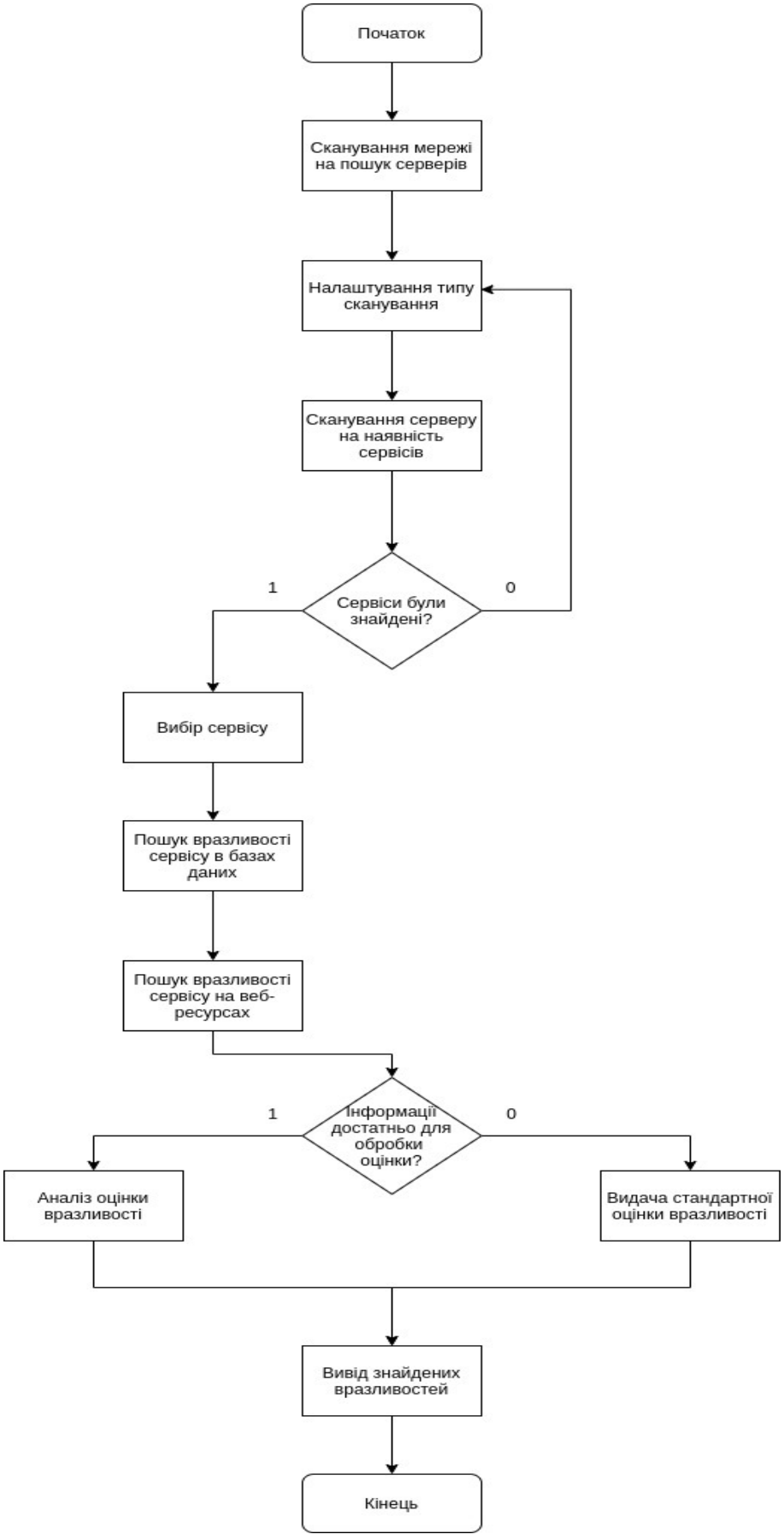
					ІАЛЦ. 467200.004 Д1	
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*		
Розробив		*Лавріненко Н.Т.*			*Система знаходження вразливостей серверу*	*Літ.* / *Аркуш* / *Аркушів*: *1* / *1*
Перевір.						
					Діаграма класів	*НТУУ “ КПІ ім. Ігоря Сікорського ” , ФІОТ, ІО-62*
Н. контр.		*Сімоненко В.П.*				
Затверд.						

Додаток 2

до дипломного проєкту

на тему: «Система знаходження вразливостей серверу»

Київ - 2020 року



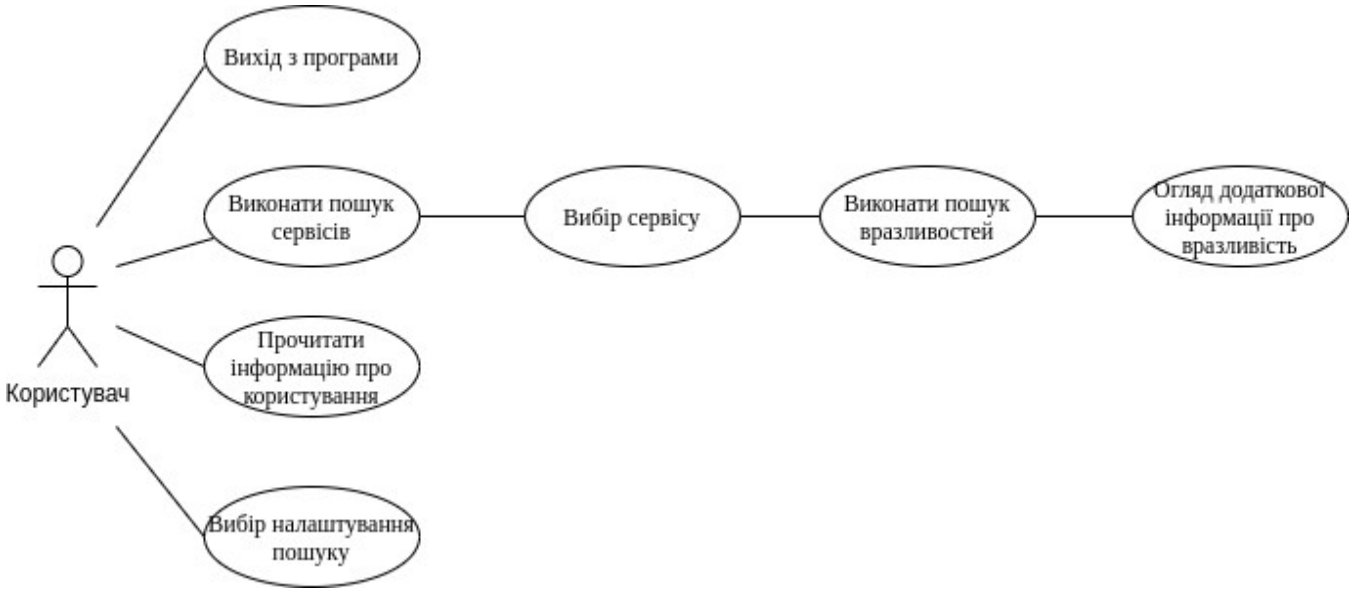
					ІАЛЦ. 467200.005 Д2		
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розробив		*Лавріненко Н.Т.*			*Система знаходження вразливостей серверу*	*Літ.*	*Аркуш* / *Аркушів*
Перевір.							*1* / *1*
Н. контр.		*Сімоненко В.П.*			Схема програми	*НТУУ “ КПІ ім. Ігоря Сікорського ” , ФІОТ, ІО-62*	
Затверд.							

Додаток 3

до дипломного проєкту

на тему: «Система знаходження вразливостей серверу»

Київ - 2020 року



					ІАЛЦ. 467200.006 *ДЗ*		
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розробив		*Лавріненко Н.Т.*			*Система знаходження вразливостей серверу*	*Літ.*	*Аркуш*	*Аркушів*
Перевір.							*1*	*1*
					Схема взаємодії програми з користувачем	*НТУУ “ КПІ ім. Ігоря Сікорського ” , ФІОТ, ІО-62*	
Н. контр.		*Сімоненко В.П.*					
Затверд.							

Додаток 4

до дипломного проєкту

на тему: «Система знаходження вразливостей серверу»

Київ - 2020 року

ТЕКСТ ПРОГРАМИ

```
class   Vuln:
    def __init__(self, name, url=None, cve=None, cve_url=None,
                        exploit=None, tp=None, verf=None, score=None):
        self.name = name
        self.cve = cve
        self.cve_url = cve_url
        self.type = tp
        self.verf = verf
        self.url = url
        self.exploit = exploit
        self.score = score

    def __str__(self):
        return str(self.cve)

    def __repr__(self):
        return str(self.cve)

import nmap
import sys
import requests
import socket
import urllib3
from bs4 import BeautifulSoup
from vuln_class import Vuln

def parse_input():
    if len(sys.argv) > 1:
        ip = sys.argv[1]
        port_range = '1-65535'
        if len(sys.argv) == 3:
            port_range = sys.argv[2]
    else:
```

```
        print("usage : python3 main.py ip port_range")
        exit(0)
    print("ip : %s port_range : %s" % (ip, port_range))
    return ip, port_range

def out_scan_info(nm, ip):
    ports = []
    for protocol in nm[ip].all_protocols():
        all_ports = list(nm[ip][protocol].keys())
        all_ports.sort()
        for port in all_ports:
            ports.append([port, nm[ip][protocol][port]['state'],
            nm[ip][protocol][port]['name'], nm[ip][protocol][port]['product'],
            nm[ip][protocol][port]['version']])
    return ports

def print_scan_info(nm, ip):
    print("Hostname : %s" % (nm[ip].hostname()))
    for protocol in nm[ip].all_protocols():
        print("~~~~~~~~~~~~")
        print("Protocol : %s" % (protocol))
        all_ports = list(nm[ip][protocol].keys())
        all_ports.sort()
        print("port\tstate\tname\t\tproduct\t\tversion")
        for port in all_ports:
            print("%s\t%s\t%s\t\t%s\t\t%s" % (port, nm[ip][protocol][port]['state'],
            nm[ip][protocol][port]['name'], nm[ip][protocol][port]['product'],
            nm[ip][protocol][port]['version']))

def exploit_db(href):
    exp_url = "www.exploit-db.com"
    url = href[7:href.find("&")]
    print(url)
    headers = {
        'user-agent': 'Mozilla/5.0 (X11; Linux i686; rv:10.0) Gecko/20100101 Firefox/10.0',
        'referrer': 'https://google.com',
```

```
        'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/
*;q=0.8',
        'Accept-Encoding': 'gzip, deflate, br',
        'Accept-Language': 'en-US,en;q=0.9',
        'Pragma': 'no-cache',
    }
    page = requests.get(url, headers=headers)
    soup = BeautifulSoup(page.content, 'html.parser')
    name = soup.find_all("h1", {"class" : "card-title"})[0].text.strip()
    cve = "CVE-" +soup.find_all("h6", {"class" : "stats-title"})[1].text.strip()
    cve_url = soup.find_all("h6")[1].find("a").get("href")
    verf = 1 if soup.find_all("i", {"class" : "mdi-check"}) else 0
    exploit = exp_url + soup.find_all("a", {"title" : "View Raw"})[0].get("href")
    tp = soup.find_all("h6", {"class" : "stats-title"})[3].text.strip()
    return Vuln(name=name, cve=cve, cve_url=cve_url,
    exploit=exploit, tp=tp, verf=verf, url=url)


def cve_details(href):
    exp_db = "www.cvedetails.com"
    url = href[7:href.find("&")]
    print(url)
    headers = {
        'user-agent': 'Mozilla/5.0 (X11; Linux i686; rv:10.0) Gecko/20100101 Firefox/10.0',
        'referrer': 'https://google.com',
        'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/
*;q=0.8',
        'Accept-Encoding': 'gzip, deflate, br',
        'Accept-Language': 'en-US,en;q=0.9',
        'Pragma': 'no-cache',
    }
    vulns = []
    http = urllib3.PoolManager()
    page = http.request('GET', url)
    soup = BeautifulSoup(page.data, 'html.parser')
    tr = soup.find_all("tr", {"class" : "srrowns"})
    descr = soup.find_all("td", {"class" : "cvesummarylong"})
    len_descr = len(descr)
```

```
    min_descr = len_descr if len_descr < 3 else 3
    for i in range(min_descr):
        score = tr[i].find_all("div", {"class" : "cvssbox"})[0].text.strip()
        name = descr[i].text.strip()
        cve = tr[i].find_all("a")[1].text
        cve_url = exp_db + "/cve/" + cve
        tp = tr[i].find_all("td")[9].text.strip()
        vulns.append(Vuln(name=name, cve_url=cve_url, cve=cve, url="http://" + cve_url,
tp=tp,score=score))
    return (vulns)



def srcap_vuln_info(name, product, version):
    exp_db = "www.exploit-db.com"
    cve_det = "cvedetails"
    page = requests.get('https://www.google.com/search?q='+
                " " + name + " " + product + " " + version + " vulnerability")
    # print('https://www.google.com/search?q='+
    #             " " + name + " " + product + " " + version + " vulnerability")
    soup = BeautifulSoup(page.content, 'html.parser')
    links = soup.find_all("a")
    vulns = []
    for link in links:
        href = link.get("href")
        if exp_db in href:
            vulns.append(exploit_db(href))
        if cve_det in href:
            vulns += cve_details(href)
    return vulns


def look_up_ports(nm, ip):
    vulns = dict()
    products = []
    for protocol in nm[ip].all_protocols():
        all_ports = list(nm[ip][protocol].keys())
        for port in all_ports:
            if (nm[ip][protocol][port]['name'] != "unknown"\
```

```
                and nm[ip][protocol][port]['name'] not in products):
                vulns[port] = srcap_vuln_info(nm[ip][protocol][port]['name'],
                        nm[ip][protocol][port]['product'],
                        nm[ip][protocol][port]['version'])
                products.append(nm[ip][protocol][port]['name'])

    print(vulns)

def legal_ip(ip):

    try:
        socket.inet_aton(ip)
        return True
    except socket.error:
        return False

def check_port(port):
    return (port < 0 or port > 65535)

def parse_kivy(ip, port_range):

    if (not legal_ip(ip)):
        ip = "127.0.0.1"
    if (not port_range):
        port_range = "1-65535"
    else:
        split_port = port_range.split('-')
        if (check_port(int(split_port[0]))):
            split_port = "1"
        if (len(split_port) == 2):
            if (check_port(int(split_port[1]))):
                split_port = "65535"
            port_range = str(split_port[0]) + '-' + str(split_port[1])
    return ip, port_range

def np_scan(ip, port_range):
```

```
    ip, port_range = parse_kivy(ip, port_range)
    nm = nmap.PortScanner()
    nm.scan(ip, port_range)

    return ip, nm, out_scan_info(nm, ip)

def start_scan(ip, port_range):

    # ip, port_range = parse_input()
    ip, port_range = parse_kivy(ip, port_range)

    nm = nmap.PortScanner()
    nm.scan(ip, port_range)

    print_scan_info(nm, ip)
    look_up_ports(nm, ip)
import threading
import webbrowser
from functools import partial

from kivy.app import App
from kivy.properties import (NumericProperty, ObjectProperty,
                             ReferenceListProperty)
from kivy.uix.behaviors import ButtonBehavior
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.button import Button
from kivy.uix.floatlayout import FloatLayout
from kivy.uix.gridlayout import GridLayout
from kivy.uix.label import Label
from kivy.uix.scrollview import ScrollView
from kivy.uix.textinput import TextInput
from kivy.uix.widget import Widget
from scrap import np_scan, srcap_vuln_info
```

```
class ButtonGrid(ButtonBehavior, GridLayout):

    def __init__(self, **kwargs):
        super(ButtonGrid, self).__init__(**kwargs)
        self.row_id = 0
        self.active = 0
        self.nm_param = []
        self.vuln = None

    def select_grid(self):
        self.color = 1, 0, 0, 1

    def on_press(self):
        if (self.active):
            self.color = 0.93, 0.93, 0.93, 1
        else:
            self.color = 0.0, 0.68, 0.71, 1
        self.active = not self.active
        print("id = %s" % str(self.row_id))

class   VulnFind(FloatLayout):

    def __init__(self, **kwargs):
        super(VulnFind, self).__init__(**kwargs)

    def thread_scan(self, btn):
        ip, nm, ports = np_scan(self.target_scan.text, self.port_range.text)
        btn.disabled = False
        btn.text = "Scan"
        self.clear_all_scan(self.grid_scan)
        self.grid_scan.rows = len(ports) + 1
        for p in range(len(ports)):
            tmp_grid = ButtonGrid(cols=5, rows=1)
            tmp_grid.row_id = p
            tmp_grid.nm_param = ports[p]
            for i in range(len(ports[p])):
```

```
                show_str = str(ports[p][i]).strip()
                show_str = show_str[0:30] + "..." if len(show_str) > 30 else show_str
                label = Label(text=show_str,
                      color=self.convert_rgb(34, 40, 49))
                if i == 4:
                   label.font_size=10
                tmp_grid.add_widget(label)
            self.grid_scan.add_widget(tmp_grid)
        self.clear_widgets()
        self.port_top.opacity = 1
        self.grid_scan.size_hint_x = 1.11
        self.grid_scan.opacity = 1
        self.btn_find.opacity = 1
        self.btn_find.size_hint_x = 0.16

    def clear_all_scan(self, grid):
        if (grid.children):
            for i in range(len(grid.children) - 1, -1, -1):
                grid.remove_widget(grid.children[i])

    def scan(self, btn):
        btn.disabled = True
        btn.text = "Scanning..."
        func = threading.Thread(target=self.thread_scan, args=(btn, ))
        func.start()

    def convert_rgb(self, r, g, b):
        return round(r/255, 2), round(g/255, 2), round(b/255, 2), 1

    def thread_find(self, btn):

        vulns = dict()
        for c in self.grid_scan.children:
            if c.active:
                vulns[c.nm_param[0]] = srcap_vuln_info(*(c.nm_param[2::]))
        #print(vulns)
```

```
btn.disabled = False
btn.text = "Find Exploit"
print(*vulns.values())
self.clear_all_scan(self.grid_find)
self.grid_find.rows = 0
for vuln in (vulns.values()):
    self.grid_find.rows += len(vuln)
#print(self.grid_find.rows)
for key in vulns:
    for cve in vulns[key]:
        tmp_grid = ButtonGrid(cols=5, rows=1)
        tmp_grid.row_id = cve
        tmp_grid.vuln = cve
        l1 = Label(text=str(key),
        color=self.convert_rgb(34, 40, 49))
        l2 = Label(text=str(cve.cve),
        color=self.convert_rgb(34, 40, 49))
        l3 = Label(text=str(cve.type),
        color=self.convert_rgb(34, 40, 49))
        l4 = Label(text=str(cve.score),
        color=self.convert_rgb(34, 40, 49))
        print(cve.score)
        show_descp = str(cve.name).strip()
        show_descp = show_descp[0:35] + "..." if len(show_descp) > 35 else show_descp
        l5 = Label(text=show_descp,
        color=self.convert_rgb(34, 40, 49),
        font_size=10)
        #l4.texture_size = l3.size[0] + 100, 40
        # l4.size = l4.texture_size
        tmp_grid.add_widget(l1)
        tmp_grid.add_widget(l2)
        tmp_grid.add_widget(l3)
        tmp_grid.add_widget(l4)
        tmp_grid.add_widget(l5)
        self.grid_find.add_widget(tmp_grid)
self.clear_widgets()
```

```
        self.vuln_top.opacity = 1
        self.grid_find.opacity = 1
        self.grid_find.size_hint_x = 1.11
        self.btn_more_info.opacity = 1
        self.btn_more_info.size_hint_x = 0.16

    def find_vuln(self, btn):
        btn.disabled = True
        btn.text = "Finding..."
        func = threading.Thread(target=self.thread_find, args=(btn, ))
        func.start()



    def clear_widgets(self):
        self.btn_about.background_color = 0.93, 0.93, 0.93, 1
        self.btn_port.background_color = 0.93, 0.93, 0.93, 1
        self.btn_vuln.background_color = 0.93, 0.93, 0.93, 1
        self.port_top.opacity = 0
        self.grid_scan.opacity = 0
        self.vuln_top.opacity = 0
        self.grid_find.opacity = 0
        self.grid_find.size_hint_x = 0
        self.grid_scan.size_hint_x = 0
        self.btn_more_info.opacity = 0
        self.btn_find.opacity = 0
        self.btn_find.size_hint_x = 0
        self.btn_more_info.size_hint_x = 0

    def vuln_info(self, btn):
        self.clear_widgets()
        btn.background_color = 1, 0, 0.93, 1
        self.vuln_top.opacity = 1
        self.grid_find.opacity = 1
        self.grid_find.size_hint_x = 1.11
        self.btn_more_info.opacity = 1
        self.btn_more_info.size_hint_x = 0.16
```

```
    def port_info(self, btn):
        self.clear_widgets()
        btn.background_color = 1, 0, 0.93, 1
        self.port_top.opacity = 1
        self.grid_scan.opacity = 1
        self.grid_scan.size_hint_x = 1.11
        self.btn_find.opacity = 1
        self.btn_find.size_hint_x = 0.16

    def about_info(self, btn):
        self.clear_widgets()
        btn.background_color = 1, 0, 0.93, 1

    def open_browser(self, btn):
        for c in self.grid_find.children:
            if c.active:
                print("url %s" % (c.vuln.url))
                webbrowser.open( c.vuln.url)

class VulnFindApp(App):
    def build(self):
        app = VulnFind()
        return app

VulnFindApp().run()
#:kivy 1.11.1

<ButtonGrid>:
    color: 0.93, 0.93, 0.93, 1
    canvas.before:
        Color:
            rgb: self.color
        Rectangle:
            size: self.size
            pos: self.pos
```

```
<VulnFind>:
    target_scan: target_scan
    btn_scan: btn_scan
    scan_box: scan_box
    port_range: port_range
    grid_scan: grid_scan
    btn_port: btn_port
    btn_vuln: btn_vuln
    btn_about: btn_about
    port_top: port_top
    vuln_top: vuln_top
    grid_find: grid_find
    btn_more_info: btn_more_info
    btn_find: btn_find

    FloatLayout:
        canvas:
            Color:
                rgb: 0.13, 0.16, 0.19, 1
            Rectangle:
                size: self.size
                pos: self.pos

    TextInput:
        id: target_scan
        hint_text: "127.0.0.1"
        size_hint: 0.18, 0.05
        pos_hint: {"center_x" : 0.35, "center_y" : 0.9}

    Label:
        font_size : 20
        text : ":"
        pos_hint: {"center_x" : 0.455, "center_y" : 0.9}

    TextInput:
```

```
        id: port_range
        hint_text: "1-65535"
        size_hint: 0.1, 0.05
        pos_hint: {"center_x" : 0.52, "center_y" : 0.9}

    Button:
        id: btn_scan
        text: "Scan"
        size_hint: 0.18, 0.06
        on_press: root.scan(btn_scan)
        pos_hint: {"center_x" : 0.7, "center_y" : 0.9}



    FloatLayout:
        id: scan_box
        size_hint: 0.9, 0.7
        pos_hint: {"center_x" : 0.5, "center_y" : 0.47}
        canvas.before:
            Color:
                rgb: 0.93, 0.93, 0.93, 1
            Rectangle:
                size: self.size
                pos: self.pos

        RelativeLayout:
            GridLayout:
                cols: 3
                rows: 1
                size_hint: 1.005, 0.05
                pos_hint: {"center_x" : 0.5545, "center_y" : 1.15}
                Button:
                    id: btn_port
                    text: "Ports"
                    on_press: root.port_info(btn_port)
                Button:
```

```
            id: btn_vuln
            text: "Vulnerabilities"
            on_press: root.vuln_info(btn_vuln)
        Button:
            id: btn_about
            text: "About"
            on_press: root.about_info(btn_about)
GridLayout:
    id: port_top
    cols: 5
    padding: 50, 0
    size_hint_x: 1.11
    pos_hint: {"top" : 1.4}
    Label:
        text: 'Port'
        color: 0.13, 0.16, 0.19, 1
    Label:
        text: 'State'
        color: 0.13, 0.16, 0.19, 1
    Label:
        text: 'Name'
        color: 0.13, 0.16, 0.19, 1
    Label:
        text: 'Product'
        color: 0.13, 0.16, 0.19, 1
    Label:
        text: 'Version'
        color: 0.13, 0.16, 0.19, 1

GridLayout:
    id: vuln_top
    cols: 6
    padding: 50, 0
    opacity: 0
    size_hint_x: 1.11
    pos_hint: {"top" : 1.4}
```

```
        Label:
            text: 'Port/Product'
            color: 0.13, 0.16, 0.19, 1
        Label:
            text: 'CVE'
            color: 0.13, 0.16, 0.19, 1
        Label:
            text: 'Type'
            color: 0.13, 0.16, 0.19, 1
        Label:
            text: 'Score'
            color: 0.13, 0.16, 0.19, 1
        Label:
            text: 'Description'
            color: 0.13, 0.16, 0.19, 1

    GridLayout:
        id: grid_find
        cols: 1
        padding: 40, 0
        top: self.height
        size_hint_x: 1.11
        height: self.minimum_height
        row_default_height: '45dp'
        row_force_default: True

    GridLayout:
        id: grid_scan
        cols: 1
        padding: 40, 0
        top: self.height
        size_hint_x: 1.11
        height: self.minimum_height
        row_default_height: '35dp'
        row_force_default: True
```

```
Button:
    id: btn_find
    text: "Find Exploit"
    size_hint: 0.16, 0.07
    pos_hint: {"center_x" : 0.5, "center_y" : 0.06}
    on_press: root.find_vuln(btn_find)

Button:
    id: btn_more_info
    text: "More Info"
    opacity: 0
    size_hint: 0.0, 0.07
    pos_hint: {"center_x" : 0.5, "center_y" : 0.06}
    on_press: root.open_browser(btn_find)
```